```
 1: ************************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9.120
 3: ************************************************************************
 4:
 5: //////////////////////////////////////////////////////////////////////////
 6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX
 7: ------------------------------------------------------------------------
 8:
 9: File Size of EXE: 23614464 V3.9.9.120 Dez 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: ***************Now the Funclist*****************************************
11: Funclist Function : 13930 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 8516  //6792 //6401 4752
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1382 //995 //
16: def head:max: maXbox7: 28.11.2014 09:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: ------------------------------------------------------------------------
20: Funclist total Size all is: 23828! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 24190
22: ASize of EXE: 23614464 (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.120: 3EFB12E5729BDAA745373C2743F9DD1E93C9295D
24:
25: ------------------------------------------------------------------------
26: ------------------------------------------------------------------------
27: //////////////////////////////////////////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *************Now the Funclist*****************
32: function  GetResStringChecked(Ident: string; const Args: array of const): string
33: Function ( Index : Longint) : Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: Function _CheckAutoResult( ResultCode : HResult) : HResult
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: Function Abs(e : Extended) : Extended;
39: Function Ackermann( const A, B : Integer) : Integer
40: Function AcquireLayoutLock : Boolean
41: Function ActionByName( const AName : string) : TWebActionItem
42: Function ACTIVEBUFFER : PCHAR
43: Function Add : TAggregate
44: function Add : TCollectionItem
45: Function Add : TColumn
46: Function Add : TComboExItem
47: Function Add : TCookie
48: Function Add : TCoolBand
49: Function Add : TFavoriteLinkItem
50: Function Add : TFileTypeItem
51: Function Add : THeaderSection
52: Function Add : THTMLTableColumn
53: Function Add : TIdEMailAddressItem
54: Function Add : TIdMessagePart
55: Function Add : TIdUserAccount
56: Function Add : TListColumn
57: Function Add : TListItem
58: Function Add : TStatusPanel
59: Function Add : TTaskDialogBaseButtonItem
60: Function Add : TWebActionItem
61: Function Add : TWorkArea
62: Function Add( AClass : TClass) : Integer
63: Function Add( AComponent : TComponent) : Integer
64: Function Add( AItem, AData : Integer) : Integer
65: Function Add( AItem, AData : Pointer) : Pointer
66: Function Add( AItem, AData : TObject) : TObject
67: Function Add( AObject : TObject) : Integer
68: Function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: Function Add( const S : WideString) : Integer
70: Function Add( Image, Mask : TBitmap) : Integer
71: Function Add( Index : LongInt; const Text : string) : LongInt
72: Function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: Function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: Function ADDCHILD : TFIELDDEF
77: Function AddChild( Index : LongInt; const Text : string) : LongInt
78: Function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: Function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
81: Function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: Function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: Function ADDFIELDDEF : TFIELDDEF
84: Function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: Function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: Function AddIcon( Image : TIcon) : Integer
87: Function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: Function ADDINDEXDEF : TINDEXDEF
89: Function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImagIdx,
    Indent:Int;Data:Ptr):TComboExItem
```

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

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

```
268: Function BooleanToInteger( const Pb : Boolean) : Integer
269: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string)
270: Function BoolToStr1(value : boolean) : string;
271: Function booltoint( aBool : Boolean) : LongInt
272: Function inttobool( aInt : LongInt) : Boolean
273: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect
274: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect)
275: Function BreakApart( BaseString, BreakString : string; StringList : TStrings) : TStrings
276: Function BrightColor( const Color : TColor; const Pct : Single) : TColor
277: Function BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
278: Function BufferRequest( Length : Integer) : TStream
279: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
280: Function Buttons : PTaskDialogButton
281: Function BytesPerScanline( PixelsPerScanline, BitsPerPixel, Alignment : Longint) : Longint
282: Function BytesToCardinal( const AValue : TIdBytes; const AIndex : Integer) : Cardinal
283: Function BytesToChar( const AValue : TIdBytes; const AIndex : Integer) : Char
284: Function BytesToInt64( const AValue : TIdBytes; const AIndex : Integer) : Int64
285: Function BytesToInteger( const AValue : TIdBytes; const AIndex : Integer) : Integer
286: Function BytesToIPv6( const AValue : TIdBytes; const AIndex : Integer) : TIdIPv6Address
287: Function BytesToShort( const AValue : TIdBytes; const AIndex : Integer) : Short
288: Function BytesToString(ABytes:TIdBytes; AStartIndex:Integer; AMaxCount:Integer): string;
289: Function BytesToStr(const Value: TBytes): String;
290: Function BytesToWord( const AValue : TIdBytes; const AIndex : Integer) : Word
291: Function ByteToBin(Int: Byte): String;
292: Function ByteToCharIndex( S : string; Index : Integer) : Integer
293: function ByteToCharIndex(const S: string; Index: Integer): Integer)
294: Function ByteToCharLen( S : string; MaxLen : Integer) : Integer
295: Function ByteToCharLen(const S: string; MaxLen: Integer): Integer)
296: Function ByteToHex( const AByte : Byte) : string
297: Function ByteToOctal( const AByte : Byte) : string
298: Function ByteType( S : string; Index : Integer) : TMbcsByteType
299: function ByteType(const S: string; Index: Integer): TMbcsByteType)
300: Function CalcTitleRect( Col : TColumn; ARow : Integer; var MasterCol : TColumn) : TRect
301: Function CalculateDFAFingerprint( oStates : TList) : integer
302: function CallTerminateProcs: Boolean)
303: function CANFOCUS:BOOLEAN
304: Function CanLoad( const Ext : string) : Boolean
305: Function CanParse( AWebRequest : TWebRequest) : Boolean
306: Function CanSave( const Ext : string) : Boolean
307: Function CanStart( cChar : char) : boolean
308: Function CaptureScreen : TBitmap;
309: Function CaptureScreen1( Rec : TRect) : TBitmap;
310: Function CardinalToFourChar( ACardinal : LongWord) : string
311: Function CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Boolean): Boolean
312: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
313: Function Ceil( const X : Extended) : Integer
314: Function Ceil16( X : Integer) : Integer
315: Function Ceil4( X : Integer) : Integer
316: Function Ceil8( X : Integer) : Integer
317: Function Ceiling( const X : Extended) : Integer
318: Function CellRect( ACol, ARow : Longint) : TRect
319: Function CelsiusToFahrenheit( const AValue : Double) : Double
320: Function CenterPoint( const Rect : TRect) : TPoint
321: function CenterPoint(const Rect: TRect): TPoint)
322: Function ChangeFileExt( FileName, Extension : string) : string
323: function ChangeFileExt(const FileName: string; const Extension: string): string)
324: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean
325: Function CharInSet( const Ch : Char; const testSet: TSysCharSet): Boolean
326: Function CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean
328: Function CharLength( S : String; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : String
330: function CharsetToIdent(Charset: Longint; var Ident: string): Boolean)
331: Function CharToBin(vChr: Char): String;
332: Function CharNext(lpsz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: Function CharToByteIndex(const S: string; Index: Integer): Integer)
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer)
337: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string);
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUniCode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckBox: string;
344: Function CheckOpen( Status : DBIResult) : Boolean
345: Function CheckPassword( const APassword : String) : Boolean
346: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
347: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
348: function CheckSynchronize(Timeout: Integer): Boolean
349: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
350: Function 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 CompToDouble( 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 ConvDec1(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 ConvInc1(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 CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
497: Function CreateColumns : TDBGridColumns
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):TParameter
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 DateTimeToUnix( const AValue : TDateTime) : Int64
570: function DateTimeToUnix(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( FileName : string; Flags : FILEOP_FLAGS) : Boolean
633: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
634: Function DelimiterPosn1(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 DescriptionToConvType1(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( p1 : 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 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
684: Function DualInputQuery(const ACapt,Prpt1,Prpt2:string;var AVal1,AVal2:string;PasswrdChar: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 EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
707: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
708: Function EndOfAMonth( 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 EnsureRange1( 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: TIFException; Param: String): String;
734: function ExceptionType: TIFException;
735: Function ExcludeTrailingBackslash( S : string) : string
736: function ExcludeTrailingBackslash(const S: string): string)
737: Function ExcludeTrailingPathDelimiter( const APath : string) : string
738: Function ExcludeTrailingPathDelimiter( S : string) : string
739: function ExcludeTrailingPathDelimiter(const S: string): string)
740: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): boolean;
741: Function ExecProc : Integer
742: Function ExecSQL : Integer
743: Function ExecSQL( ExecDirect : Boolean) : Integer
744: Function Execute : _Recordset;
745: Function Execute : Boolean
746: Function Execute : Boolean;
747: Function Execute( const SQL : string; Params : TParams; Cache : Boolean; cursor : phDBICur) : Integer
748: Function Execute( const SQL : WideString; Params : TParams; ResultSet : TPSResult) : Integer
749: Function Execute( ParentWnd : HWND) : Boolean
750: Function Execute1(constCommText:WideString;const CType:TCommType;const
     ExecuteOpts:TExecuteOpts):_Recordset;
751: Function Execute1( const Parameters : OleVariant) : _Recordset;
752: Function Execute1( ParentWnd : HWND) : Boolean;
753: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant) : _Recordset;
754: Function ExecuteAction( Action : TBasicAction) : Boolean
755: Function ExecuteDirect( const SQL : WideString) : Integer
756: Function ExecuteFile(const FileName:string;const Params:string;const DefDir:string;ShowCmd:Int):THandle
757: Procedure ExecuteThread2(afunc:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure
758: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle
759: function ExeFileIsRunning(ExeFile: string): boolean;
760: function ExePath: string;
761: function ExePathName: string;
762: Function Exists( AItem : Pointer) : Boolean
763: Function ExitWindows( ExitCode : Cardinal) : Boolean
764: function Exp(x: Extended): Extended;
765: Function ExpandEnvironmentVar( var Value : string) : Boolean
766: Function ExpandFileName( FileName : string) : string
767: function ExpandFileName(const FileName: string): string)
768: Function ExpandUNCFileName( FileName : string) : string
769: function ExpandUNCFileName(const FileName: string): string)
770: Function ExpJ( const X : Float) : Float;
771: Function Exsecans( X : Float) : Float
772: Function Extract( const AByteCount : Integer) : string
773: Function Extract( Item : TClass) : TClass
774: Function Extract( Item : TComponent) : TComponent
775: Function Extract( Item : TObject) : TObject
776: Function ExtractFileDir( FileName : string) : string
777: function ExtractFileDir(const FileName: string): string)
778: Function ExtractFileDrive( FileName : string) : string
779: function ExtractFileDrive(const FileName: string): string)
780: Function ExtractFileExt( FileName : string) : string
781: function ExtractFileExt(const FileName: string): string)
782: Function ExtractFileExtNoDot( const FileName : string) : string
783: Function ExtractFileExtNoDotUpper( const FileName : string) : string
784: Function ExtractFileName( FileName : string) : string
785: function ExtractFileName(const filename: string):string;
786: Function ExtractFilePath( FileName : string) : string
787: function ExtractFilePath(const filename: string):string;
788: Function ExtractRelativePath( BaseName, DestName : string) : string
```

```
789: function ExtractRelativePath(const BaseName: string; const DestName: string): string)
790: Function ExtractShortPathName( FileName : string) : string
791: function ExtractShortPathName(const FileName: string): string)
792: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;Strings: TStrings): Integer
793: function ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings): Integer)
794: Function Fact(numb: integer): Extended;
795: Function FactInt(numb: integer): int64;
796: Function Factorial( const N : Integer) : Extended
797: Function FahrenheitToCelsius( const AValue : Double) : Double
798: function FalseBoolStrs: array of string
799: Function Fetch(var AInput:string;const ADelim:string;const ADelete:Bool;const ACaseSensitive:Bool):string
800: Function FetchCaseInsensitive(var AInput:string; const ADelim:string; const ADelete:Boolean): string
801: Function Fibo(numb: integer): Extended;
802: Function FiboInt(numb: integer): Int64;
803: Function Fibonacci( const N : Integer) : Integer
804: Function FIELDBYNAME( const FIELDNAME : STRING) : TFIELD
805: Function FIELDBYNAME( const FIELDNAME : WIDESTRING) : TFIELD
806: Function FIELDBYNAME( const NAME : String) : TFIELD
807: Function FIELDBYNAME( const NAME : String) : TFIELDDEF
808: Function FIELDBYNUMBER( FIELDNO : INTEGER) : TFIELD
809: Function FileAge( FileName : string) : Integer
810: Function FileAge(const FileName: string): integer)
811: Function FileCompareText( const A, B : String) : Integer
812: Function FileContains(const FileName:string;Text:string;CaseSensitive:Bool;ExceptionOnError:Bool): Boolean
813: Function FileCreate( FileName : string) : Integer;
814: Function FileCreate(const FileName: string): integer)
815: Function FileCreateTemp( var Prefix : string) : THandle
816: Function FileDateToDateTime( FileDate : Integer) : TDateTime
817: function FileDateToDateTime(FileDate: Integer): TDateTime;
818: Function FileExists( const FileName : string) : Boolean
819: Function FileExists( FileName : string) : Boolean
820: function fileExists(const FileName: string): Boolean;
821: Function FileGetAttr( FileName : string) : Integer
822: Function FileGetAttr(const FileName: string): integer)
823: Function FileGetDate( Handle : Integer) : Integer
824: Function FileGetDate(handle: integer): integer
825: Function FileGetDisplayName( const FileName : string) : string
826: Function FileGetSize( const FileName : string) : Integer
827: Function FileGetTempName( const Prefix : string) : string
828: Function FileGetTypeName( 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) : TIdAuthenticationClass
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,
     sfStartMenu, stStartUp, sfTemplates);
884:  FFolder: array [TJvSpecialFolder] of Integer =
885:     (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
886:      CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
887:      CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
888:      CSIDL_STARTUP, CSIDL_TEMPLATES);
889: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean);
890: function Findfirst(const filepath: string; attr: integer): integer;
891: function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer)
892: Function FindFirstNotOf( AFind, AText : String) : Integer
893: Function FindFirstOf( AFind, AText : String) : Integer
894: Function FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
895: Function FINDINDEXFORFIELDS( const FIELDS : String) : TINDEXDEF
896: Function FindInstanceOf( AClass : TClass; AExact : Boolean; AStartAt : Integer) : Integer
897: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND) : TMENUITEM
898: function FindItemId( Id : Integer) : TCollectionItem
899: Function FindKey( const KeyValues : array of const) : Boolean
900: Function FINDLAST : BOOLEAN
901: Function FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
902: Function FindModuleClass( AClass : TComponentClass) : TComponent
903: Function FindModuleName( const AClass : string) : TComponent
904: Function FINDNEXT : BOOLEAN
905: function FindNext: integer;
906: function FindNext2(var F: TSearchRec): Integer)
907: Function FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible : Boolean) : TTabSheet
908: Function FindNextToSelect : TTreeNode
909: Function FINDPARAM( const VALUE : String) : TPARAM
910: Function FindParam( const Value : WideString) : TParameter
911: Function FINDPRIOR : BOOLEAN
912: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar) : TResourceHandle
913: Function FindSession( const SessionName : string) : TSession
914: function FindStringResource(Ident: Integer): string)
915: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer
916: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString) : AnsiString
917: function FindVCLWindow(const Pos: TPoint): TWinControl;
918: function FindWindow(C1, C2: PChar): Longint;
919: Function FindInPaths(const fileName,paths: String): String;
920: Function Finger : String
921: Function First : TClass
922: Function First : TComponent
923: Function First : TObject
924: Function FirstDelimiter( const delimiters : string; const Str : String) : integer;
925: Function FirstDelimiter1( const delimiters : WideString; const Str : WideString) : integer;
926: Function FirstInstance( const ATitle : string) : Boolean
927: Function FloatPoint( const X, Y : Float) : TFloatPoint;
928: Function FloatPoint1( const P : TPoint) : TFloatPoint;
929: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) : Boolean
930: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
931: Function FloatRect1( const Rect : TRect) : TFloatRect;
932: Function FloatsEqual( const X, Y : Float) : Boolean
933: Function FloatToBin(const D: Double): string;     //doubletohex -> hextobin! in buffer
934: Function FloatToCurr( Value : Extended) : Currency
935: Function FloatToDateTime( Value : Extended) : TDateTime
936: Function FloatToStr( Value : Extended) : string;
937: Function FloatToStr(e : Extended) : String;
938: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer) : string;
939: function FloatToStrF(Value: Extended; Format: TFloatFormat;Precision:Integer; Digits: Integer): string)
940: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings
     : TFormatSettings) : string;
941: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer;
     FormatSettings : TFormatSettings) : string;
942: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue;Format: TFloatFormat;
     Precision,Digits: Integer): 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 FTPGMTDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 988: Function FTPLocalDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 989: Function FTPMLSToGMTDateTime( const ATimeStamp : String) : TDateTime
 990: Function FTPMLSToLocalDateTime( 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 : TIdHeaderList
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 GetFileInformation( 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 GetFileSize2(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 : LongInt
1129: Function getFirstChild : TTreeNode
1130: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string) : string
1131: Function GetFirstNode : TTreeNode
1132: Function GetFontsFolder : string
1133: Function GetFormulaValue( const Formula : string) : Extended
1134: Function GetFreePageFileMemory : Integer
1135: Function GetFreePhysicalMemory : Integer
1136: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind) : Integer;
1137: Function GetFreeSystemResources1 : TFreeSystemResources;
1138: Function GetFreeVirtualMemory : Integer
```

```
1139: Function GetFromClipboard : Boolean
1140: Function GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet) : String
1141: Function GetGBitmap( Value : TBitmap) : TBitmap
1142: Function GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
1143: Function GetGroupState( Level : Integer) : TGroupPosInds
1144: Function GetHandle : HWND
1145: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN) : THELPCONTEXT
1146: Function GetHexArray(ahexdig: THexArray): THexArray;
1147: Function GetHighLightColor( const Color : TColor; Luminance : Integer) : TColor
1148: function GetHINSTANCE: longword;
1149: Function GetHistoryFolder : string
1150: Function GetHitTestInfoAt( X, Y : Integer) : THitTests
1151: function getHMODULE: longword;
1152: Function GetHostByName(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 GetIPHostByName(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: TStrings;
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 : IDENTIFIEROption) : 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 HexStrToStr(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( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1527: Function iif3( ATest : 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 APath : string) : string
1548: Function IncludeTrailingPathDelimiter( S : string) : string
1549: function IncludeTrailingPathDelimiter(const S: string): string)
1550: Function IncludeTrailingSlash( const APath : 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: TIdIPv6Address): 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 ISqrt( const I : Smallint) : Smallint
1713: Function IsReadOnly(const Filename: string): boolean;
1714: Function IsRectEmpty( const Rect : TRect) : Boolean
1715: function IsRectEmpty(const Rect: TRect): 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 IsWoW64: 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 MakeDIB( out Bitmap : PBitmapInfo) : Integer
1846: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1847: Function MakeFile(const FileName: string): integer)');
1848: function MakeLong(A, B: Word): Longint)
1849: Function MakeTempFilename( const APath : 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 MaskIntlLiteralToChar( 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 AText : string; const AValues : array of string) : Boolean
1871: Function MatchText( const AText : 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 MD5(const fileName: string): string;
1892: Function Mean( const Data : array of Double) : Extended
1893: Function Median( const X : TDynFloatArray) : Float
1894: Function Memory : Pointer
1895: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1896: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1897: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1898: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1899: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1900: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1901: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1902: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1903: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1904: Function MibToId( Mib : string) : string
1905: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1906: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1907: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1908: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1909: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1910: Procedure GetMidiOutputs( const List : TStrings)
1911: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
1912: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
1913: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1914: Function MIDINoteToStr( Note : TMIDINote) : string
1915: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1916: Procedure GetMidiOutputs( const List : TStrings)
1917: Procedure MidiOutCheck( Code : MMResult)
1918: Procedure MidiInCheck( Code : MMResult)
1919: Function MilliSecondOf( const AValue : TDateTime) : Word
1920: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1921: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1922: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1923: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1924: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1925: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
```

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

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

```
2098: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2099: Function Peek : Pointer
2100: Function Peek : TObject
2101: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2102: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2103: function Permutation(npr, k: integer): extended;
2104: function PermutationInt(npr, k: integer): Int64;
2105: Function PermutationJ( N, R : Cardinal) : Float
2106: Function Pi : Extended;
2107: Function PiE : Extended;
2108: Function PixelsToDialogsX( const Pixels : Word) : Word
2109: Function PixelsToDialogsY( const Pixels : Word) : Word
2110: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2111: Function Point( X, Y : Integer) : TPoint
2112: function Point(X, Y: Integer): TPoint)
2113: Function PointAssign( const X, Y : Integer) : TPoint
2114: Function PointDist( const P1, P2 : TPoint) : Double;
2115: function PointDist(const P1,P2: TFloatPoint): Double;
2116: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2117: function PointDist2(const P1,P2: TPoint): Double;
2118: Function PointEqual( const P1, P2 : TPoint) : Boolean
2119: Function PointIsNull( const P : TPoint) : Boolean
2120: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2121: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2122: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2123: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2124: Function Pop : Pointer
2125: Function Pop : TObject
2126: Function PopnStdDev( const Data : array of Double) : Extended
2127: Function PopnVariance( const Data : array of Double) : Extended
2128: Function PopulationVariance( const X : TDynFloatArray) : Float
2129: function Pos(SubStr, S: AnyString): Longint;
2130: Function PosEqual( const Rect : TRect) : Boolean
2131: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2132: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2133: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2134: Function Post1( AURL : string; const ASource : TStrings) : string;
2135: Function Post2( AURL : string; const ASource : TStream) : string;
2136: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2137: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2138: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2139: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2140: Function Power( const Base, Exponent : Extended) : Extended
2141: Function PowerBig(aval, n:integer): string;
2142: Function PowerIntJ( const X : Float; N : Integer) : Float;
2143: Function PowerJ( const Base, Exponent : Float) : Float;
2144: Function PowerOffOS : Boolean
2145: Function PreformatDateString( Ps : string) : string
2146: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2147: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2148: Function Printer : TPrinter
2149: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2150: Function ProcessResponse : TIdHTTPWhatsNext
2151: Function ProduceContent : string
2152: Function ProduceContentFromStream( Stream : TStream) : string
2153: Function ProduceContentFromString( const S : string) : string
2154: Function ProgIDToClassID(const ProgID: string): TGUID;
2155: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2156: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2157: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2158: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2159: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2160: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2161: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2162: Function Push( AItem : Pointer) : Pointer
2163: Function Push( AObject : TObject) : TObject
2164: Function Put1( AURL : string; const ASource : TStream) : string;
2165: Function Pythagoras( const X, Y : Extended) : Extended
2166: Function queryDLLInterface( var queryList : TStringList) : TStringList
2167: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2168: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2169: Function queryPerformanceCounter2(mse: int64): int64;
2170: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2171: //Function QueryPerformanceFrequency(mse: int64): boolean;
2172: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2173: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2174: Procedure QueryPerformanceCounter1(var aC: Int64);
2175: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2176: Function Quote( const ACommand : String) : SmallInt
2177: Function QuotedStr( S : string) : string
2178: Function RadToCycle( const Radians : Extended) : Extended
2179: Function RadToDeg( const Radians : Extended) : Extended
2180: Function RadToDeg( const Value : Extended) : Extended;
2181: Function RadToDeg1( const Value : Double) : Double;
```

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

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

```
2357: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2358: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2359: Function RunByteCode(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2360: Function RunCompiledScript2(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2361: Function S_AddBackSlash( const ADirName : string) : string
2362: Function S_AllTrim( const cStr : string) : string
2363: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2364: Function S_Cut( const cStr : string; const iLen : integer) : string
2365: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2366: Function S_DirExists( const ADir : string) : Boolean
2367: Function S_Empty( const cStr : string) : boolean
2368: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2369: Function S_LargeFontsActive : Boolean
2370: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2371: Function S_LTrim( const cStr : string) : string
2372: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2373: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2374: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2375: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2376: Function S_RTrim( const cStr : string) : string
2377: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2378: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2379: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2380: Function S_Space( const iLen : integer) : String
2381: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2382: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2383: Function S_StrCRC32( const Text : string) : LongWORD
2384: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2385: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2386: Function S_StringtoUTF_8( const AString : string) : string
2387: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2388: function S_StrToReal(const cStr: string; var R: Double): Boolean
2389: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2390: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2391: Function S_UTF_8ToString( const AString : string) : string
2392: Function S_WBox( const AText : string) : integer
2393: Function SameDate( const A, B : TDateTime) : Boolean
2394: function SameDate(const A, B: TDateTime): Boolean;
2395: Function SameDateTime( const A, B : TDateTime) : Boolean
2396: function SameDateTime(const A, B: TDateTime): Boolean;
2397: Function SameFileName( S1, S2 : string) : Boolean
2398: Function SameText( S1, S2 : string) : Boolean
2399: function SameText(const S1: string; const S2: string): Boolean)
2400: Function SameTime( const A, B : TDateTime) : Boolean
2401: function SameTime(const A, B: TDateTime): Boolean;
2402: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2403: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2404: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2405: Function SampleVariance( const X : TDynFloatArray) : Float
2406: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2407: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2408: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2409: Function SaveToFile( const AFileName : TFileName) : Boolean
2410: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2411: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2412: Function ScanF(const aformat: String; const args: array of const): string;
2413: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2414: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2415: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2416: function SearchRecattr: integer;
2417: function SearchRecExcludeAttr: integer;
2418: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2419: function SearchRecname: string;
2420: function SearchRecsize: integer;
2421: function SearchRecTime: integer;
2422: Function Sec( const X : Extended) : Extended
2423: Function Secant( const X : Extended) : Extended
2424: Function SecH( const X : Extended) : Extended
2425: Function SecondOf( const AValue : TDateTime) : Word
2426: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2427: Function SecondOfTheHour( const AValue : TDateTime) : Word
2428: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2429: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2430: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2431: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2432: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2433: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2434: Function SectionExists( const Section : string) : Boolean
2435: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2436: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2437: function Seek(Offset:LongInt;Origin:Word):LongInt
2438: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2439: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2440: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2441: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2442: Function SendBuf( var Buf, Count : Integer) : Integer
```

```
2443: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2444: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2445: Function SendKey( AppName : string; Key : Char) : Boolean
2446: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2447: Function SendStream( AStream : TStream) : Boolean
2448: Function SendStreamThenDrop( AStream : TStream) : Boolean
2449: Function SendText( const S : string) : Integer
2450: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2451: Function SendSerialText(Data: String): cardinal
2452: Function Sent : Boolean
2453: Function ServicesFilePath: string
2454: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2455: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2456: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2457: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2458: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2459: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2460: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2461: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2462: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2463: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2464: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2465: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2466: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2467: Function SetCurrentDir( Dir : string) : Boolean
2468: function SetCurrentDir(const Dir: string): Boolean)
2469: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2470: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2471: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2472: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2473: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2474: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2475: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2476: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2477: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2478: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2479: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2480: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2481: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2482: Function SetLocalTime( Value : TDateTime) : boolean
2483: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2484: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2485: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2486: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2487: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2488: Function SetSize( libNewSize : Longint) : HResult
2489: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2490: Function Sgn( const X : Extended) : Integer
2491: function SHA1(const fileName: string): string;
2492: function SHA256(astr: string; amode: char): string)
2493: function SHA512(astr: string; amode: char): string)
2494: Function ShareMemoryManager : Boolean
2495: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2496: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2497: function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2498: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2499: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2500: function ShortDateFormat: string;
2501: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2502: function ShortTimeFormat: string;
2503: function SHOWMODAL:INTEGER
2504: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
      TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult');
2505: Function
      ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2506: function ShowWindow(C1: HWND; C2: integer): boolean;
2507: procedure ShowMemory     //in Dialog
2508: function ShowMemory2: string;
2509: Function ShutDownOS : Boolean
2510: Function Signe( const X, Y : Extended) : Extended
2511: Function Sign( const X : Extended) : Integer
2512: Function Sin(e : Extended) : Extended;
2513: Function sinc( const x : Double) : Double
2514: Function SinJ( X : Float) : Float
2515: Function Size( const AFileName : String) : Integer
2516: function SizeOf: Longint;
2517: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2518: function SlashSep(const Path, S: String): String
2519: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2520: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2521: Function SmallPoint(X, Y: Integer): TSmallPoint)
2522: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2523: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2524: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2525: Function SoundexProc( const AText, AOther : string) : Boolean
2526: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2527: Function SoundexWord( const AText : string) : Word
2528: Function SourcePos : Longint
```

```
2529: function SourcePos:LongInt
2530: Function Split0( Str : string; const substr : string) : TStringList
2531: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2532: Function SQLRequiresParams( const SQL : WideString) : Boolean
2533: Function Sqr(e : Extended) : Extended;
2534: Function Sqrt(e : Extended) : Extended;
2535: Function StartIP : String
2536: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2537: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2538: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2539: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2540: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2541: Function StartOfAYear( const AYear : Word) : TDateTime
2542: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2543: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2544: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2545: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2546: Function StartsStr( const ASubText, AText : string) : Boolean
2547: Function StartsText( const ASubText, AText : string) : Boolean
2548: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2549: Function StartsWith( const str : string; const sub : string) : Boolean
2550: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2551: Function StatusString( StatusCode : Integer) : string
2552: Function StdDev( const Data : array of Double) : Extended
2553: Function Stop : Float
2554: Function StopCount( var Counter : TJclCounter) : Float
2555: Function StoreColumns : Boolean
2556: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2557: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2558: Function StrAlloc( Size : Cardinal) : PChar
2559: function StrAlloc(Size: Cardinal): PChar)
2560: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2561: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2562: Function StrBufSize( Str : PChar) : Cardinal
2563: function StrBufSize(const Str: PChar): Cardinal)
2564: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2565: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2566: Function StrCat( Dest : PChar; Source : PChar) : PChar
2567: function StrCat(Dest: PChar; const Source: PChar): PChar)
2568: Function StrCharLength( Str : PChar) : Integer
2569: Function StrComp( Str1, Str2 : PChar) : Integer
2570: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2571: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2572: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2573: Function Stream_to_AnsiString( Source : TStream) : ansistring
2574: Function Stream_to_Base64( Source : TStream) : ansistring
2575: Function Stream_to_decimalbytes( Source : TStream) : string
2576: Function Stream2WideString( oStream : TStream) : WideString
2577: Function StreamtoAnsiString( Source : TStream) : ansistring
2578: Function StreamToByte( Source : TStream) : string
2579: Function StreamToDecimalbytes( Source : TStream) : string
2580: Function StreamtoOrd( Source : TStream) : string
2581: Function StreamToString( Source : TStream) : string
2582: Function StreamToString2( Source : TStream) : string
2583: Function StreamToString3( Source : TStream) : string
2584: Function StreamToString4( Source : TStream) : string
2585: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2586: Function StrEmpty( const sString : string) : boolean
2587: Function StrEnd( Str : PChar) : PChar
2588: function StrEnd(const Str: PChar): PChar)
2589: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2590: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2591: Function StrGet(var S : String; I : Integer) : Char;
2592: Function StrGet2(S : String; I : Integer) : Char;
2593: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2594: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2595: Function StrHtmlDecode( const AStr : String) : String
2596: Function StrHtmlEncode( const AStr : String) : String
2597: Function StrToBytes(const Value: String): TBytes;
2598: Function StrIComp( Str1, Str2 : PChar) : Integer
2599: Function StringOfChar(c : char;I : longInt) : String;
2600: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2601: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2602: Function StringRefCount(const s: String): integer;
2603: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2604: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2605: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2606: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2607: Function StringToBoolean( const Ps : string) : Boolean
2608: function StringToColor(const S: string): TColor)
2609: function StringToCursor(const S: string): TCursor;
2610: function StringToGUID(const S: string): TGUID)
2611: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2612: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2613: Function StringWidth( S : string) : Integer
2614: Function StrInternetToDateTime( Value : string) : TDateTime
2615: Function StrIsDateTime( const Ps : string) : Boolean
2616: Function StrIsFloatMoney( const Ps : string) : Boolean
2617: Function StrIsInteger( const S : string) : Boolean
```

```
2618: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2619: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2620: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2621: Function StrLen( Str : PChar) : Cardinal
2622: function StrLen(const Str: PChar): Cardinal)
2623: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2624: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2625: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2626: Function StrLower( Str : PChar) : PChar
2627: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2628: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2629: Function StrNew( Str : PChar) : PChar
2630: function StrNew(const Str: PChar): PChar)
2631: Function StrNextChar( Str : PChar) : PChar
2632: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2633: Function StrParse( var sString : string; const sDelimiters : string) : string;
2634: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2635: Function StrPas( Str : PChar) : string
2636: function StrPas(const Str: PChar): string)
2637: Function StrPCopy( Dest : PChar; Source : string) : PChar
2638: function StrPCopy(Dest: PChar; const Source: string): PChar)
2639: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2640: Function StrPos( Str1, Str2 : PChar) : PChar
2641: Function StrScan(const Str: PChar; Chr: Char): PChar)
2642: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2643: Function StrToBcd( const AValue : string) : TBcd
2644: Function StrToBool( S : string) : Boolean
2645: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2646: Function StrToCard( const AStr : String) : Cardinal
2647: Function StrToConv( AText : string; out AType : TConvType) : Double
2648: Function StrToCurr( S : string) : Currency;
2649: function StrToCurr(const S: string): Currency)
2650: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2651: Function StrToDate( S : string) : TDateTime;
2652: function StrToDate(const s: string): TDateTime;
2653: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2654: Function StrToDateTime( S : string) : TDateTime;
2655: function StrToDateTime(const S: string): TDateTime)
2656: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2657: Function StrToDay( const ADay : string) : Byte
2658: Function StrToFloat( S : string) : Extended;
2659: function StrToFloat(s: String): Extended;
2660: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2661: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2662: Function StrToFloat( S : string) : Extended;
2663: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2664: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2665: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2666: Function StrToCurr( S : string) : Currency;
2667: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2668: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2669: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2670: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2671: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2672: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2673: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2674: Function StrToDateTime( S : string) : TDateTime;
2675: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2676: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2677: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2678: Function StrToInt( S : string) : Integer
2679: function StrToInt(s: String): Longint;
2680: Function StrToInt64( S : string) : Int64
2681: function StrToInt64(s: String): int64;
2682: Function StrToInt64Def( S : string; Default : Int64) : Int64
2683: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2684: Function StrToIntDef( S : string; Default : Integer) : Integer
2685: function StrToIntDef(const S: string; Default: Integer): Integer)
2686: function StrToIntDef(s: String; def: Longint): Longint;
2687: Function StrToMonth( const AMonth : string) : Byte
2688: Function StrToTime( S : string) : TDateTime;
2689: function StrToTime(const S: string): TDateTime)
2690: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2691: Function StrToWord( const Value : String) : Word
2692: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2693: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2694: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2695: Function StrUpper( Str : PChar) : PChar
2696: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2697: Function Sum( const Data : array of Double) : Extended
2698: Function SumFloatArray( const B : TDynFloatArray) : Float
2699: Function SumInt( const Data : array of Integer) : Integer
2700: Function SumOfSquares( const Data : array of Double) : Extended
2701: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2702: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2703: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2704: Function Supports( CursorOptions : TCursorOptions) : Boolean
2705: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2706: Function SwapWord(w : word): word)
```

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

```
2790: Function Trace2( AURL : string) : string;
2791: Function TrackMenu( Button : TToolButton) : Boolean
2792: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2793: Function TranslateURI( const URI : string) : string
2794: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2795: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2796: Function Trim( S : string) : string;
2797: Function Trim( S : WideString) : WideString;
2798: Function Trim(s : AnyString) : AnyString;
2799: Function TrimAllOf( ATrim, AText : String) : String
2800: Function TrimLeft( S : string) : string;
2801: Function TrimLeft( S : WideString) : WideString;
2802: function TrimLeft(const S: string): string)
2803: Function TrimRight( S : string) : string;
2804: Function TrimRight( S : WideString) : WideString;
2805: function TrimRight(const S: string): string)
2806: function TrueBoolStrs: array of string
2807: Function Trunc(e : Extended) : Longint;
2808: Function Trunc64(e: extended): Int64;
2809: Function TruncPower( const Base, Exponent : Float) : Float
2810: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2811: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2812: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2813: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2814: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2815: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
2816: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2817: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2818: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2819: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2820: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2821: Function TryLock : Boolean
2822: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2823: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2824: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2825: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2826: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2827: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2828: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2829: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2830: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2831: function TryStrToBool(const S: string; out Value: Boolean): Boolean;
2832: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2833: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2834: Function TwoToY( const Y : Float) : Float
2835: Function UCS4StringToWideString( const S : UCS4String) : WideString
2836: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2837: function Unassigned: Variant;
2838: Function UndoLastChange( FollowChange : Boolean) : Boolean
2839: function UniCodeToStr(Value: string): string;
2840: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2841: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2842: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2843: Function UnixPathToDosPath( const Path : string) : string
2844: Function UnixToDateTime( const AValue : Int64) : TDateTime
2845: function UnixToDateTime(U: Int64): TDateTime;
2846: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2847: Function UnlockResource( ResData : HGLOBAL) : LongBool
2848: Function UnlockVolume( var Handle : THandle) : Boolean
2849: Function UnMaskString( Mask, Value : String) : String
2850: function UpCase(ch : Char ) : Char;
2851: Function UpCaseFirst( const AStr : string) : string
2852: Function UpCaseFirstWord( const AStr : string) : string
2853: Function UpdateAction( Action : TBasicAction) : Boolean
2854: Function UpdateKind : TUpdateKind
2855: Function UPDATESTATUS : TUPDATESTATUS
2856: Function UpperCase( S : string) : string
2857: Function Uppercase(s : AnyString) : AnyString;
2858: Function URLDecode( ASrc : string) : string
2859: Function URLEncode( const ASrc : string) : string
2860: Function UseRightToLeftAlignment : Boolean
2861: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2862: Function UseRightToLeftReading : Boolean
2863: Function UTF8CharLength( Lead : Char) : Integer
2864: Function UTF8CharSize( Lead : Char) : Integer
2865: Function UTF8Decode( const S : UTF8String) : WideString
2866: Function UTF8Encode( const WS : WideString) : UTF8String
2867: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2868: Function Utf8ToAnsi( const S : UTF8String) : string
2869: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2870: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2871: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2872: Function ValidParentForm(control: TControl): TForm
2873: Function Value : Variant
2874: Function ValueExists( const Section, Ident : string) : Boolean
```

```
2875: Function ValueOf( const Key : string) : Integer
2876: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2877: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2878: Function VarArrayFromStrings( Strings : TStrings) : Variant
2879: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2880: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2881: Function VarFMTBcd : TVarType
2882: Function VarFMTBcdCreate1 : Variant;
2883: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2884: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2885: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2886: Function Variance( const Data : array of Double) : Extended
2887: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2888: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2889: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2890: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2891: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2892: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2893: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2894: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2895: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2896: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2897: Function VariantNeg( const V1 : Variant) : Variant
2898: Function VariantNot( const V1 : Variant) : Variant
2899: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2900: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2901: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2902: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2903: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2904: function VarIsEmpty(const V: Variant): Boolean;
2905: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2906: function VarIsNull(const V: Variant): Boolean;
2907: Function VarToBcd( const AValue : Variant) : TBcd
2908: function VarType(const V: Variant): TVarType;
2909: Function VarType( const V : Variant) : TVarType
2910: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2911: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2912: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2913: Function VarIsByRef( const V : Variant) : Boolean
2914: Function VarIsEmpty( const V : Variant) : Boolean
2915: Procedure VarCheckEmpty( const V : Variant)
2916: Function VarIsNull( const V : Variant) : Boolean
2917: Function VarIsClear( const V : Variant) : Boolean
2918: Function VarIsCustom( const V : Variant) : Boolean
2919: Function VarIsOrdinal( const V : Variant) : Boolean
2920: Function VarIsFloat( const V : Variant) : Boolean
2921: Function VarIsNumeric( const V : Variant) : Boolean
2922: Function VarIsStr( const V : Variant) : Boolean
2923: Function VarToStr( const V : Variant) : string
2924: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2925: Function VarToWideStr( const V : Variant) : WideString
2926: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2927: Function VarToDateTime( const V : Variant) : TDateTime
2928: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2929: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2930: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2931: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2932: Function VarSameValue( const A, B : Variant) : Boolean
2933: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2934: Function VarIsEmptyParam( const V : Variant) : Boolean
2935: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2936: Function VarIsError1( const V : Variant) : Boolean;
2937: Function VarAsError( AResult : HRESULT) : Variant
2938: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2939: Function VarIsArray( const A : Variant) : Boolean;
2940: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2941: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2942: Function VarArrayOf( const Values : array of Variant) : Variant
2943: Function VarArrayRef( const A : Variant) : Variant
2944: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2945: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2946: Function VarArrayDimCount( const A : Variant) : Integer
2947: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2948: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2949: Function VarArrayLock( const A : Variant) : ___Pointer
2950: Procedure VarArrayUnlock( const A : Variant)
2951: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2952: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2953: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2954: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2955: Function Unassigned : Variant
2956: Function Null : Variant
2957: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2958: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2959: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2960: function VectorDot(const V1,V2: TFloatPoint): Double;
2961: Function VectorLengthSqr( const V : TFloatPoint) : Double
2962: function VectorLengthSqr(const V: TFloatPoint): Double;
2963: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
```

```
2964: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2965: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2966: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2967: Function Verify( AUserName : String) : String
2968: Function Versine( X : Float) : Float
2969: function VersionCheck: boolean;
2970: function VersionCheckAct: string;
2971: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2972: Function VersionLanguageName( const LangId : Word) : string
2973: Function VersionResourceAvailable( const FileName : string) : Boolean
2974: Function Visible : Boolean
2975: function VolumeID(DriveChar: Char): string
2976: Function WaitFor( const AString : string) : string
2977: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2978: Function WaitFor1 : TWaitResult;
2979: Function WaitForData( Timeout : Longint) : Boolean
2980: Function WebColorNameToColor( WebColorName : string) : TColor
2981: Function WebColorStrToColor( WebColor : string) : TColor
2982: Function WebColorToRGB( WebColor : Integer) : Integer
2983: function wGet(aURL, afile: string): boolean;'
2984: Function wGet2(aURL, afile: string): boolean;'  //without file open
2985: Function wGetX(aURL, afile: string): boolean;
2986: Function wGetX2(aURL, afile: string): boolean;'  //without file open
2987: Function WebGet(aURL, afile: string): boolean;'
2988: Function WebExists: boolean;   //alias to isinternet
2989: Function WeekOf( const AValue : TDateTime) : Word
2990: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2991: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2992: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2993: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2994: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2995: Function WeeksInAYear( const AYear : Word) : Word
2996: Function WeeksInYear( const AValue : TDateTime) : Word
2997: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2998: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2999: Function WideCat( const x, y : WideString) : WideString
3000: Function WideCompareStr( S1, S2 : WideString) : Integer
3001: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
3002: Function WideCompareText( S1, S2 : WideString) : Integer
3003: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
3004: Function WideCopy( const src : WideString; index, count : Integer) : WideString
3005: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
3006: Function WideEqual( const x, y : WideString) : Boolean
3007: function WideFormat(const Format: WideString; const Args: array of const): WideString)
3008: Function WideGreater( const x, y : WideString) : Boolean
3009: Function WideLength( const src : WideString) : Integer
3010: Function WideLess( const x, y : WideString) : Boolean
3011: Function WideLowerCase( S : WideString) : WideString
3012: function WideLowerCase(const S: WideString): WideString)
3013: Function WidePos( const src, sub : WideString) : Integer
3014: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
3015: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
3016: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
3017: Function WideSameStr( S1, S2 : WideString) : Boolean
3018: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
3019: Function WideSameText( S1, S2 : WideString) : Boolean
3020: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
3021: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
3022: Function WideStringToUCS4String( const S : WideString) : UCS4String
3023: Function WideUpperCase( S : WideString) : WideString
3024: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
3025: function Win32Check(RetVal: boolean): boolean)
3026: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
3027: Function Win32RestoreFile( const FileName : string) : Boolean
3028: Function Win32Type : TIdWin32Type
3029: Function WinColor( const Color32 : TColor32) : TColor
3030: function winexec(FileName: pchar; showCmd: integer): integer;
3031: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3032: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3033: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
3034: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
3035: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
3036: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
3037: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
3038: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
3039: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
3040: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
3041: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
3042: Function WordToStr( const Value : Word) : String
3043: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
3044: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
3045: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
3046: Function WorkArea : Integer
3047: Function WrapText( Line : string; MaxCol : Integer) : string;
3048: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
3049: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3050: function Write(Buffer:String;Count:LongInt):LongInt
3051: Function WriteClient( var Buffer, Count : Integer) : Integer
3052: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
```

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

```
3140: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3141: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3142: Function DialogsToPixelsX( Dlgs : Word) : Word
3143: Function DialogsToPixelsY( Dlgs : Word) : Word
3144: Function PixelsToDialogsX( Pixs : Word) : Word
3145: Function PixelsToDialogsY( Pixs : Word) : Word
3146: {$IFDEF WIN32}
3147: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3148: function MakeVariant(const Values: array of Variant): Variant;
3149: {$ENDIF}
3150: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3151: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3152: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3153: {$IFDEF CBUILDER}
3154: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3155: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3156: {$ELSE}
3157: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3158: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3159: {$ENDIF CBUILDER}
3160: function IsForegroundTask: Boolean;
3161: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3162: function GetAveCharSize(Canvas: TCanvas): TPoint;
3163: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3164: procedure FreeUnusedOle;
3165: procedure Beep;
3166: function GetWindowsVersionJ: string;
3167: function LoadDLL(const LibName: string): THandle;
3168: function RegisterServer(const ModuleName: string): Boolean;
3169: {$IFNDEF WIN32}
3170: function IsLibrary: Boolean;
3171: {$ENDIF}
3172: { Gradient filling routine }
3173: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3174: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
      TFillDirection; Colors: Byte);
3175: { String routines }
3176: function GetEnvVar(const VarName: string): string;
3177: function AnsiUpperFirstChar(const S: string): string;
3178: function StringToPChar(var S: string): PChar;
3179: function StrPAlloc(const S: string): PChar;
3180: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3181: function DropT(const S: string): string;
3182: { Memory routines }
3183: function AllocMemo(Size: Longint): Pointer;
3184: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3185: procedure FreeMemo(var fpBlock: Pointer);
3186: function GetMemoSize(fpBlock: Pointer): Longint;
3187: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3188: {$IFNDEF COMPILER5_UP}
3189: procedure FreeAndNil(var Obj);
3190: {$ENDIF}
3191: // from PNGLoader
3192: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3193: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3194: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3195: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3196: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
      TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3197:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3198:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3199:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3200:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3201:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3202:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3203:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3204:  Procedure SetImeName( Name : TImeName)
3205:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3206:  Function Imm32GetContext( hWnd : HWND) : HIMC
3207:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3208:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3209:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3210:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3211: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3212:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3213:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3214:  Function Imm32IsIME( hKl : longword) : Boolean
3215:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3216:  Procedure DragDone( Drop : Boolean)
3217:
3218:
3219: //**************************************added  from jvjvclutils
3220: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3221: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3222: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3223: function IsPositiveResult(Value: TModalResult): Boolean;
3224: function IsNegativeResult(Value: TModalResult): Boolean;
3225: function IsAbortResult(const Value: TModalResult): Boolean;
3226: function StripAllFromResult(const Value: TModalResult): TModalResult;
```

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

```
3314:  Const('sdfStandard16','String '''"''mm''/''dd''/''yyyy''"''
3315:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3316:  Const('sdfOracle','String '"TO_DATE(''"dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3317:  Const('sdfInterbase','String '"CAST(''"mm"/"dd"/"yyyy"'' AS DATE)"
3318:  Const('sdfMSSQL','String '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3319:  AddTypeS('Largeint', 'Longint
3320:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3321:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3322:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3323:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3324:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3325: (*----------------------------------------------------------------------------*)
3326: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3327: begin
3328:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3329:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3330:  Function JIniReadString( const FileName, Section, Line : string) : string
3331:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3332:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3333:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3334:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3335:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3336: end;
3337:
3338: (* === compile-time registration functions === *)
3339: (*----------------------------------------------------------------------------*)
3340: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3341: begin
3342:  'UnixTimeStart','LongInt'( 25569);
3343:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3344:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3345:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3346:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3347:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3348:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3349:  Function DayOfDate( const DateTime : TDateTime) : Integer
3350:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3351:  Function YearOfDate( const DateTime : TDateTime) : Integer
3352:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3353:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3354:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3355:  Function HourOfTime( const DateTime : TDateTime) : Integer
3356:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3357:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3358:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3359:  Function IsISOLongYear( const Year : Word) : Boolean;
3360:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3361:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3362:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3363:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3364:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3365:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3366:  Function JIsLeapYear( const Year : Integer) : Boolean;
3367:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3368:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3369:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3370:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3371:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3372:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3373:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3374:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3375:  Function HoursToMSecs( Hours : Integer) : Integer
3376:  Function MinutesToMSecs( Minutes : Integer) : Integer
3377:  Function SecondsToMSecs( Seconds : Integer) : Integer
3378:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3379:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3380:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3381:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3382:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3383:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3384:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3385:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3386:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3387:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3388:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3389:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3390:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3391:  Function DosDateTimeToStr( DateTime : Integer) : string
3392:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3393:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3394:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3395:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3396:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3397:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3398:  Function FileTimeToStr( const FileTime : TFileTime) : string
3399:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3400:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3401:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3402:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
```

```
3403:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3404:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3405:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3406:   TJclUnixTime32', 'Longword
3407:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3408:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3409:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3410:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3411:  Function JNullStamp : TTimeStamp
3412:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3413:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3414:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3415:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3416:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3417:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3418:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3419:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3420:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3421:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3422:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3423:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3424:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3425:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3426:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3427:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3428:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3429:   FindClass('TOBJECT'),'EJclDateTimeError
3430: end;
3431:
3432: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3433: begin
3434:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3435:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3436:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3437:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3438:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3439:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3440:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3441:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3442:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3443:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3444:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3445:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3446:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3447:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3448:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3449:  Function AbortShutDown : Boolean;
3450:  Function AbortShutDown1( const MachineName : string) : Boolean;
3451:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3452:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3453:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3454:   FindClass('TOBJECT'),'EJclCreateProcessError
3455:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3456:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
      Environment:PChar);
3457:   // with Add(EJclCreateProcessError) do
3458:  end;
3459:
3460:
3461: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3462: begin
3463:   //'AnsiSigns','Set').SetSet(['-', '+']);
3464:  'C1_UPPER','LongWord( $0001);
3465:  'C1_LOWER','LongWord( $0002);
3466:  'C1_DIGIT','LongWord').SetUInt( $0004);
3467:  'C1_SPACE','LongWord').SetUInt( $0008);
3468:  'C1_PUNCT','LongWord').SetUInt( $0010);
3469:  'C1_CNTRL','LongWord').SetUInt( $0020);
3470:  'C1_BLANK','LongWord').SetUInt( $0040);
3471:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3472:  'C1_ALPHA','LongWord').SetUInt( $0100);
3473:   AnsiChar', 'Char
3474:  Function StrIsAlpha( const S : AnsiString) : Boolean
3475:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3476:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3477:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3478:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3479:  Function StrIsDigit( const S : AnsiString) : Boolean
3480:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3481:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3482:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3483:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3484:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3485:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3486:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3487:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3488:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3489:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3490:  Function StrEscapedToString( const S : AnsiString) : AnsiString
```

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

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

```
3666: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3667: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3668: Procedure DbNotSupported
3669: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3670: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3671: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3672: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3673: ******************************************uPSI_JvDateUtil;
3674: function CurrentYear: Word;
3675: function IsLeapYear(AYear: Integer): Boolean;
3676: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3677: function FirstDayOfPrevMonth: TDateTime;
3678: function LastDayOfPrevMonth: TDateTime;
3679: function FirstDayOfNextMonth: TDateTime;
3680: function ExtractDay(ADate: TDateTime): Word;
3681: function ExtractMonth(ADate: TDateTime): Word;
3682: function ExtractYear(ADate: TDateTime): Word;
3683: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3684: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3685: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3686: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3687: function ValidDate(ADate: TDateTime): Boolean;
3688: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3689: function MonthsBetween(Date1, Date2: TDateTime): Double;
3690: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3691: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3692: function DaysBetween(Date1, Date2: TDateTime): Longint;
3693: { The same as previous but if Date2 < Date1 result = 0 }
3694: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3695: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3696: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3697: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3698: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3699: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3700: { String to date conversions }
3701: function GetDateOrder(const DateFormat: string): TDateOrder;
3702: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3703: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3704: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3705: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3706: function DefDateFormat(FourDigitYear: Boolean): string;
3707: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3708: ----------------------------------------------------------------------------
3709: ******************************** JvUtils;*****************************
3710: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3711: function GetWordOnPos(const S: string; const P: Integer): string;
3712: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3713: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3714: { SubStr returns substring from string, S, separated with Separator string}
3715: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3716: { SubStrEnd same to previous function but Index numerated from the end of string }
3717: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3718: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3719: function SubWord(P: PChar; var P2: PChar): string;
3720: { NumberByWord returns the text representation of
3721:   the number, N, in normal russian language. Was typed from Monitor magazine }
3722: function NumberByWord(const N: Longint): string;
3723: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3724:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3725: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3726: { GetXYByPos is same to previous function, but returns X position in line too}
3727: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3728: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3729: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3730: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3731: function ConcatSep(const S, S2, Separator: string): string;
3732: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3733: function ConcatLeftSep(const S, S2, Separator: string): string;
3734: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3735: function MinimizeString(const S: string; const MaxLen: Integer): string;
3736: { Next 4 function for russian chars transliterating.
3737:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3738: procedure Dos2Win(var S: string);
3739: procedure Win2Dos(var S: string);
3740: function Dos2WinRes(const S: string): string;
3741: function Win2DosRes(const S: string): string;
3742: function Win2Koi(const S: string): string;
3743: { Spaces returns string consists on N space chars }
3744: function Spaces(const N: Integer): string;
3745: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3746: function AddSpaces(const S: string; const N: Integer): string;
3747: { function LastDate for russian users only } { returns date relative to current date: '' }
3748: function LastDate(const Dat: TDateTime): string;
3749: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3750: function CurrencyToStr(const Cur: currency): string;
3751: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
```

```
3752: function Cmp(const S1, S2: string): Boolean;
3753: { StringCat add S2 string to S1 and returns this string }
3754: function StringCat(var S1: string; S2: string): string;
3755: { HasChar returns True, if Char, Ch, contains in string, S }
3756: function HasChar(const Ch: Char; const S: string): Boolean;
3757: function HasAnyChar(const Chars: string; const S: string): Boolean;
3758: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3759: function CountOfChar(const Ch: Char; const S: string): Integer;
3760: function DefStr(const S: string; Default: string): string;
3761: {**** files routines}
3762: { GetWinDir returns Windows folder name }
3763: function GetWinDir: TFileName;
3764: function GetSysDir: String;
3765: { GetTempDir returns Windows temporary folder name }
3766: function GetTempDir: string;
3767: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3768: function GenTempFileName(FileName: string): string;
3769: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3770: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3771: { ClearDir clears folder Dir }
3772: function ClearDir(const Dir: string): Boolean;
3773: { DeleteDir clears and than delete folder Dir }
3774: function DeleteDir(const Dir: string): Boolean;
3775: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3776: function FileEquMask(FileName, Mask: TFileName): Boolean;
3777: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3778:   Masks must be separated with comma (';') }
3779: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3780: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3781: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3782: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3783: { FileGetInfo fills SearchRec record for specified file attributes}
3784: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3785: { HasSubFolder returns True, if folder APath contains other folders }
3786: function HasSubFolder(APath: TFileName): Boolean;
3787: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3788: function IsEmptyFolder(APath: TFileName): Boolean;
3789: { AddSlash add slash Char to Dir parameter, if needed }
3790: procedure AddSlash(var Dir: TFileName);
3791: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3792: function AddSlash2(const Dir: TFileName): string;
3793: { AddPath returns FileName with Path, if FileName not contain any path }
3794: function AddPath(const FileName, Path: TFileName): TFileName;
3795: function AddPaths(const PathList, Path: string): string;
3796: function ParentPath(const Path: TFileName): TFileName;
3797: function FindInPath(const FileName, PathList: string): TFileName;
3798: function FindInPaths(const fileName,paths: String): String;
3799: {$IFNDEF BCB1}
3800: { BrowseForFolder displays Browse For Folder dialog }
3801: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3802: {$ENDIF BCB1}
3803: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3804: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3805: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3806: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3807:
3808: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3809: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3810: { HasParam returns True, if program running with specified parameter, Param }
3811: function HasParam(const Param: string): Boolean;
3812: function HasSwitch(const Param: string): Boolean;
3813: function Switch(const Param: string): string;
3814: { ExePath returns ExtractFilePath(ParamStr(0)) }
3815: function ExePath: TFileName;
3816: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3817: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3818: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3819: {**** Graphic routines }
3820: { TTFontSelected returns True, if True Type font is selected in specified device context }
3821: function TTFontSelected(const DC: HDC): Boolean;
3822: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3823: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3824: {**** Windows routines }
3825: { SetWindowTop put window to top without recreating window }
3826: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3827: {**** other routines }
3828: { KeyPressed returns True, if Key VK is now pressed }
3829: function KeyPressed(VK: Integer): Boolean;
3830: procedure SwapInt(var Int1, Int2: Integer);
3831: function IntPower(Base, Exponent: Integer): Integer;
3832: function ChangeTopException(E: TObject): TObject;
3833: function StrToBool(const S: string): Boolean;
3834: {$IFNDEF COMPILER3_UP}
3835: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3836:   Length of MaxLen bytes. The compare operation is controlled by the
3837:   current Windows locale. The return value is the same as for CompareStr. }
3838: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
```

```
3839: function AnsiStrIComp(S1, S2: PChar): Integer;
3840: {$ENDIF}
3841: function Var2Type(V: Variant; const VarType: Integer): Variant;
3842: function VarToInt(V: Variant): Integer;
3843: function VarToFloat(V: Variant): Double;
3844: { following functions are not documented because they are don't work properly , so don't use them }
3845: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3846: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3847: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3848: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3849: function GetParameter: string;
3850: function GetLongFileName(FileName: string): string;
3851: {* from  FileCtrl}
3852: function DirectoryExists(const Name: string): Boolean;
3853: procedure ForceDirectories(Dir: string);
3854: {# from  FileCtrl}
3855: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3856: function GetComputerID: string;
3857: function GetComputerName: string;
3858: {**** string routines }
3859: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 function }
3860: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3861: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3862:   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 }
3863: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3864: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3865: function CountOfLines(const S: string): Integer;
3866: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3867: procedure DeleteEmptyLines(Ss: TStrings);
3868: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3869:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3870: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3871: {**** files routines - }
3872: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3873:   Resource can be compressed using MS Compress program}
3874: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3875: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3876: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3877: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3878: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3879: { IniReadSection read section, Section, from ini-file,
3880:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3881:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3882: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3883: { LoadTextFile load text file, FileName, into string }
3884: function LoadTextFile(const FileName: TFileName): string;
3885: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3886: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3887: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3888: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3889: {$IFDEF COMPILER3_UP}
3890: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3891: function TargetFileName(const FileName: TFileName): TFileName;
3892: { return filename ShortCut linked to }
3893: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3894: {$ENDIF COMPILER3_UP}
3895: {**** Graphic routines - }
3896: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3897: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3898: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3899: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3900: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3901: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3902: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3903: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3904: { Cinema draws some visual effect }
3905: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3906: { Roughed fills rect with special 3D pattern }
3907: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3908: { 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 }
3909: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3910: { TextWidth calculate text with for writing using standard desktop font }
3911: function TextWidth(AStr: string): Integer;
3912: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3913: function DefineCursor(Identifer: PChar): TCursor;
3914: {**** other routines - }
3915: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3916: function FindFormByClass(FormClass: TFormClass): TForm;
3917: function FindFormByClassName(FormClassName: string): TForm;
3918: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3919:   having Tag property value, equaled to Tag parameter }
3920: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3921: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3922: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3923: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3924: function RBTag(Parent: TWinControl): Integer;
```

```
3925: { AppMinimized returns True, if Application is minimized }
3926: function AppMinimized: Boolean;
3927: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3928:   if Caption parameter = '', it replaced with Application.Title }
3929: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3930: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3931:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3932: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3933:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3934: { Delay stop program execution to MSec msec }
3935: procedure Delay(MSec: Longword);
3936: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3937: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3938: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3939: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3940: function PanelBorder(Panel: TCustomPanel): Integer;
3941: function Pixels(Control: TControl; APixels: Integer): Integer;
3942: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3943: procedure Error(const Msg: string);
3944: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3945:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3946:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3947: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3948:   const HideSelColor: Boolean): string;
3949: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3950:   const HideSelColor: Boolean): Integer;
3951: function ItemHtPlain(const Text: string): string;
3952: { ClearList - clears list of TObject }
3953: procedure ClearList(List: TList);
3954: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3955: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3956: { RTTI support }
3957: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3958: function GetPropStr(Obj: TObject; const PropName: string): string;
3959: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3960: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3961: procedure PrepareIniSection(SS: TStrings);
3962: { following functions are not documented because they are don't work properly, so don't use them }
3963: {$IFDEF COMPILER2}
3964: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3965: {$ENDIF}
3966:
3967: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3968: begin
3969:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3970:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3971:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3972:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3973:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3974:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3975:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3976:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3977: end;
3978:
3979: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3980: begin
3981:  Const('MaxInitStrNum','LongInt'( 9);
3982: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3983: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3984: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
      QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3985: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3986: Function JvStrStrip( S : string) : string
3987: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3988: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3989: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3990: Function StrEatWhiteSpace( const S : string) : string
3991: Function HexToAscii( const S : AnsiString) : AnsiString
3992: Function AsciiToHex( const S : AnsiString) : AnsiString
3993: Function StripQuotes( const S1 : AnsiString) : AnsiString
3994: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3995: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3996: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3997: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3998: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3999: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
4000: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
4001: Function JvValidIdentifier( S1 : String) : Boolean
4002: Function JvEndChar( X : AnsiChar) : Boolean
4003: Procedure JvGetToken( S1, S2 : PAnsiChar)
4004: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
4005: Function IsKeyword( S1 : PAnsiChar) : Boolean
4006: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
4007: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
4008: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
4009: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
```

```
4010: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
4011: Function GetTokenCount : Integer
4012: Procedure ResetTokenCount
4013: end;
4014:
4015: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
4016: begin
4017:   SIRegister_TJvQueryParamsDialog(CL);
4018:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
4019: end;
4020:
4021: ******************************** JvStrUtil /  JvStrUtils;****************************
4022: function FindNotBlankCharPos(const S: string): Integer;
4023: function AnsiChangeCase(const S: string): string;
4024: function GetWordOnPos(const S: string; const P: Integer): string;
4025: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4026: function Cmp(const S1, S2: string): Boolean;
4027: { Spaces returns string consists on N space chars }
4028: function Spaces(const N: Integer): string;
4029: { HasChar returns True, if char, Ch, contains in string, S }
4030: function HasChar(const Ch: Char; const S: string): Boolean;
4031: function HasAnyChar(const Chars: string; const S: string): Boolean;
4032: { SubStr returns substring from string, S, separated with Separator string}
4033: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4034: { SubStrEnd same to previous function but Index numerated from the end of string }
4035: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4036: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4037: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
4038: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4039: { GetXYByPos is same to previous function, but returns X position in line too}
4040: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4041: { AddSlash returns string with added slash char to Dir parameter, if needed }
4042: function AddSlash2(const Dir: TFileName): string;
4043: { AddPath returns FileName with Path, if FileName not contain any path }
4044: function AddPath(const FileName, Path: TFileName): TFileName;
4045: { ExePath returns ExtractFilePath(ParamStr(0)) }
4046: function ExePath: TFileName;
4047: function LoadTextFile(const FileName: TFileName): string;
4048: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4049: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4050: function ConcatSep(const S, S2, Separator: string): string;
4051: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4052: function FileEquMask(FileName, Mask: TFileName): Boolean;
4053: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4054:   Masks must be separated with comma (';') }
4055: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4056: function StringEndsWith(const Str, SubStr: string): Boolean;
4057: function ExtractFilePath2(const FileName: string): string;
4058: function StrToOem(const AnsiStr: string): string;
4059: { StrToOem translates a string from the Windows character set into the OEM character set. }
4060: function OemToAnsiStr(const OemStr: string): string;
4061: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4062: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4063: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4064: function ReplaceStr(const S, Srch, Replace: string): string;
4065: { Returns string with every occurrence of Srch string replaced with Replace string. }
4066: function DelSpace(const S: string): string;
4067: { DelSpace return a string with all white spaces removed. }
4068: function DelChars(const S: string; Chr: Char): string;
4069: { DelChars return a string with all Chr characters removed. }
4070: function DelBSpace(const S: string): string;
4071: { DelBSpace trims leading spaces from the given string. }
4072: function DelESpace(const S: string): string;
4073: { DelESpace trims trailing spaces from the given string. }
4074: function DelRSpace(const S: string): string;
4075: { DelRSpace trims leading and trailing spaces from the given string. }
4076: function DelSpace1(const S: string): string;
4077: { DelSpace1 return a string with all non-single white spaces removed. }
4078: function Tab2Space(const S: string; Numb: Byte): string;
4079: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4080: function NPos(const C: string; S: string; N: Integer): Integer;
4081: { NPos searches for a N-th position of substring C in a given string. }
4082: function MakeStr(C: Char; N: Integer): string;
4083: function MS(C: Char; N: Integer): string;
4084: { MakeStr return a string of length N filled with character C. }
4085: function AddChar(C: Char; const S: string; N: Integer): string;
4086: { AddChar return a string left-padded to length N with characters C. }
4087: function AddCharR(C: Char; const S: string; N: Integer): string;
4088: { AddCharR return a string right-padded to length N with characters C. }
4089: function LeftStr(const S: string; N: Integer): string;
4090: { LeftStr return a string right-padded to length N with blanks. }
4091: function RightStr(const S: string; N: Integer): string;
4092: { RightStr return a string left-padded to length N with blanks. }
4093: function CenterStr(const S: string; Len: Integer): string;
4094: { CenterStr centers the characters in the string based upon the Len specified. }
4095: function CompStr(const S1, S2: string): Integer;
4096: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4097: function CompText(const S1, S2: string): Integer;
4098: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
```

```
4099: function Copy2Symb(const S: string; Symb: Char): string;
4100: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4101: function Copy2SymbDel(var S: string; Symb: Char): string;
4102: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4103: function Copy2Space(const S: string): string;
4104: { Copy2Symb returns a substring of a string S from begining to first white space. }
4105: function Copy2SpaceDel(var S: string): string;
4106: { Copy2SpaceDel returns a substring of a string S from begining to first
4107:   white space and removes this substring from S. }
4108: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4109: { Returns string, with the first letter of each word in uppercase,
4110:   all other letters in lowercase. Words are delimited by WordDelims. }
4111: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4112: { WordCount given a set of word delimiters, returns number of words in S. }
4113: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4114: { Given a set of word delimiters, returns start position of N'th word in S. }
4115: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4116: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4117: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4118: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4119:   delimiters, return the N'th word in S. }
4120: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4121: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4122: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4123: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4124: function QuotedString(const S: string; Quote: Char): string;
4125: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4126: function ExtractQuotedString(const S: string; Quote: Char): string;
4127: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4128:    and reduces pairs of Quote characters within the quoted string to a single character. }
4129: function FindPart(const HelpWilds, InputStr: string): Integer;
4130: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4131: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4132: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4133: function XorString(const Key, Src: ShortString): ShortString;
4134: function XorEncode(const Key, Source: string): string;
4135: function XorDecode(const Key, Source: string): string;
4136: { ** Command line routines ** }
4137: {$IFNDEF COMPILER4_UP}
4138: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4139: {$ENDIF}
4140: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4141: { ** Numeric string handling routines ** }
4142: function Numb2USA(const S: string): string;
4143: { Numb2USA converts numeric string S to USA-format. }
4144: function Dec2Hex(N: Longint; A: Byte): string;
4145: function D2H(N: Longint; A: Byte): string;
4146: { Dec2Hex converts the given value to a hexadecimal string representation
4147:   with the minimum number of digits (A) specified. }
4148: function Hex2Dec(const S: string): Longint;
4149: function H2D(const S: string): Longint;
4150: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4151: function Dec2Numb(N: Longint; A, B: Byte): string;
4152: { Dec2Numb converts the given value to a string representation with the
4153:   base equal to B and with the minimum number of digits (A) specified. }
4154: function Numb2Dec(S: string; B: Byte): Longint;
4155: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4156: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4157: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4158: function IntToRoman(Value: Longint): string;
4159: { IntToRoman converts the given value to a roman numeric string representation. }
4160: function RomanToInt(const S: string): Longint;
4161: { RomanToInt converts the given string to an integer value. If the string
4162:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4163: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4164: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4165: ********************************* JvFileUtil;*********************************
4166: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4167: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4168: procedure MoveFile(const FileName, DestName: TFileName);
4169: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4170: {$IFDEF COMPILER4_UP}
4171: function GetFileSize(const FileName: string): Int64;
4172: {$ELSE}
4173: function GetFileSize(const FileName: string): Longint;
4174: {$ENDIF}
4175: function FileDateTime(const FileName: string): TDateTime;
4176: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4177: function DeleteFiles(const FileMask: string): Boolean;
4178: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4179: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4180: function NormalDir(const DirName: string): string;
4181: function RemoveBackSlash(const DirName: string): string;
4182: function ValidFileName(const FileName: string): Boolean;
4183: function DirExists(Name: string): Boolean;
4184: procedure ForceDirectories(Dir: string);
4185: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4186:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
```

```
4187: {$IFDEF COMPILER4_UP}
4188: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4189: {$ENDIF}
4190: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4191: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4192: {$IFDEF COMPILER4_UP}
4193: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4194: {$ENDIF}
4195: function GetTempDir: string;
4196: function GetWindowsDir: string;
4197: function GetSystemDir: string;
4198: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4199: {$IFDEF WIN32}
4200: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4201: function ShortToLongFileName(const ShortName: string): string;
4202: function ShortToLongPath(const ShortName: string): string;
4203: function LongToShortFileName(const LongName: string): string;
4204: function LongToShortPath(const LongName: string): string;
4205: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4206: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4207: {$ENDIF WIN32}
4208: {$IFNDEF COMPILER3_UP}
4209: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4210: {$ENDIF}
4211: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4212: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4213: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4214: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4215:
4216: //**************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4217:  Procedure VariantClear( var V : Variant)
4218:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4219:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4220:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4221:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4222:  Procedure VariantSub( const src : Variant; var dst : Variant)
4223:  Procedure VariantMul( const src : Variant; var dst : Variant)
4224:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4225:  Procedure VariantMod( const src : Variant; var dst : Variant)
4226:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4227:  Procedure VariantOr( const src : Variant; var dst : Variant)
4228:  Procedure VariantXor( const src : Variant; var dst : Variant)
4229:  Procedure VariantShl( const src : Variant; var dst : Variant)
4230:  Procedure VariantShr( const src : Variant; var dst : Variant)
4231:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4232:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4233:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4234:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4235:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4236:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4237:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4238:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4239:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4240:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4241:  Function VariantNot( const V1 : Variant) : Variant
4242:  Function VariantNeg( const V1 : Variant) : Variant
4243:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4244:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4245:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4246:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4247:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4248:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4249:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4250:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4251:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4252:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4253: end;
4254:
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; ADelineated, 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);
```

```
4275: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4276:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4277: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4278: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4279: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4280:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4281:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4282:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4283: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4284:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4285:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4286:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4287: procedure BringParentWindowToTop(Wnd: TWinControl);
4288: function GetParentForm(Control: TControl): TForm;
4289: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4290: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4291: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4292: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4293: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4294: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4295: function CalcMathString(AExpression: string): Single;
4296: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4297: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4298: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4299: procedure TypeStringOnKeyboard(const S: string);
4300: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4301: procedure DrawTextExtAligned(Canvas:TCanvas;const
     Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4302: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4303: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4304: function ComponentToString(Component: TComponent): string;
4305: procedure StringToComponent(Component: TComponent; const Value: string);
4306: function PlayWaveResource(const ResName: string): Boolean;
4307: function UserName: string;
4308: function ComputerName: string;
4309: function CreateIniFileName: string;
4310: function ExpandString(const Str: string; Len: Integer): string;
4311: function Transliterate(const Str: string; RusToLat: Boolean): string;
4312: function IsSmallFonts: Boolean;
4313: function SystemColorDepth: Integer;
4314: function GetFileTypeJ(const FileName: string): TglFileType;
4315: Function GetFileType( hFile : THandle) : DWORD');
4316: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4317: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4318:
4319: { ************************************Utility routines of unit classes}
4320: function LineStart(Buffer, BufPos: PChar): PChar
4321: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4322:   'Strings: TStrings): Integer
4323:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4324:  Procedure RegisterClass( AClass : TPersistentClass)
4325:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4326:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4327:  Procedure UnRegisterClass( AClass : TPersistentClass)
4328:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4329:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4330:  Function FindGlobalComponent( const Name : string) : TComponent
4331:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4332:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4333:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4334:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4335:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4336:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4337:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4338:  Procedure GlobalFixupReferences
4339:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4340:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4341:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4342:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4343:  Procedure RemoveFixups( Instance : TPersistent)
4344:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4345:  Procedure BeginGlobalLoading
4346:  Procedure NotifyGlobalLoading
4347:  Procedure EndGlobalLoading
4348:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4349:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4350:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4351:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4352:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4353: // Function AllocateHWnd( Method : TWndMethod) : HWND
4354:  Procedure DeallocateHWnd( Wnd : HWND)
4355:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4356: ************************************unit uPSI_SqlTimSt and DB;************************************
4357: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4358: Function VarSQLTimeStampCreate3: Variant;
4359: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4360: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4361: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4362: Function VarSQLTimeStamp : TVarType
```

```
4363: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4364: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4365: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4366: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4367: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4368: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4369: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4370: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4371: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4372: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4373: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4374: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4375: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4376:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4377: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4378: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4379: Procedure DisposeMem( var Buffer, Size : Integer)
4380: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4381: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4382: Function VarTypeToDataType( VarType : Integer) : TFieldType
4383: ********************************************unit JvStrings;****************************************
4384: {template functions}
4385: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4386: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4387: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4388: function RemoveMasterBlocks(const SourceStr: string): string;
4389: function RemoveFields(const SourceStr: string): string;
4390: {http functions}
4391: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4392: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4393: {set functions}
4394: procedure SplitSet(AText: string; AList: TStringList);
4395: function JoinSet(AList: TStringList): string;
4396: function FirstOfSet(const AText: string): string;
4397: function LastOfSet(const AText: string): string;
4398: function CountOfSet(const AText: string): Integer;
4399: function SetRotateRight(const AText: string): string;
4400: function SetRotateLeft(const AText: string): string;
4401: function SetPick(const AText: string; AIndex: Integer): string;
4402: function SetSort(const AText: string): string;
4403: function SetUnion(const Set1, Set2: string): string;
4404: function SetIntersect(const Set1, Set2: string): string;
4405: function SetExclude(const Set1, Set2: string): string;
4406: {replace any <,> etc by &lt; &gt;}
4407: function XMLSafe(const AText: string): string;
4408: {simple hash, Result can be used in Encrypt}
4409: function Hash(const AText: string): Integer;
4410: { Base64 encode and decode a string }
4411: function B64Encode(const S: AnsiString): AnsiString;
4412: function B64Decode(const S: AnsiString): AnsiString;
4413: {Basic encryption from a Borland Example}
4414: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4415: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4416: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4417: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4418: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4419: procedure CSVToTags(Src, Dst: TStringList);
4420: // converts a csv list to a tagged string list
4421: procedure TagsToCSV(Src, Dst: TStringList);
4422: // converts a tagged string list to a csv list
4423: // only fieldnames from the first record are scanned ib the other records
4424: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4425: {selects akey=avalue from Src and returns recordset in Dst}
4426: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4427: {filters Src for akey=avalue}
4428: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4429: {orders a tagged Src list by akey}
4430: function PosStr(const FindString, SourceString: string;
4431:   StartPos: Integer = 1): Integer;
4432: { PosStr searches the first occurrence of a substring FindString in a string
4433:   given by SourceString with case sensitivity (upper and lower case characters
4434:   are differed). This function returns the index value of the first character
4435:   of a specified substring from which it occurs in a given string starting with
4436:   StartPos character index. If a specified substring is not found Q_PosStr
4437:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4438: function PosStrLast(const FindString, SourceString: string): Integer;
4439: {finds the last occurance}
4440: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4441: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4442: { PosText searches the first occurrence of a substring FindString in a string
4443:   given by SourceString without case sensitivity (upper and lower case characters are not differed). This
      function returns the index value of the first character of a specified substring from which it occurs in a
      given string starting with Start
4444: function PosTextLast(const FindString, SourceString: string): Integer;
4445: {finds the last occurance}
4446: function NameValuesToXML(const AText: string): string;
4447: {$IFDEF MSWINDOWS}
4448: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4449: {$ENDIF MSWINDOWS}
```

```
4450: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4451: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4452: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4453: procedure SaveString(const AFile, AText: string);
4454: Procedure SaveStringasFile( const AFile, AText : string)
4455: function LoadStringJ(const AFile: string): string;
4456: Function LoadStringofFile( const AFile : string) : string
4457: Procedure SaveStringtoFile( const AFile, AText : string)
4458: Function LoadStringfromFile( const AFile : string) : string
4459: function HexToColor(const AText: string): TColor;
4460: function UppercaseHTMLTags(const AText: string): string;
4461: function LowercaseHTMLTags(const AText: string): string;
4462: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4463: function RelativePath(const ASrc, ADst: string): string;
4464: function GetToken(var Start: Integer; const SourceText: string): string;
4465: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4466: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4467: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4468: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4469: // parses the beginning of an attribute: space + alpha character
4470: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4471: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4472: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4473: // parses all name=value attributes to the attributes TStringList
4474: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4475: // checks if a name="value" pair exists and returns any value
4476: function GetStrValue(const AText, AName, ADefault: string): string;
4477: // retrieves string value from a line like:
4478: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4479: // returns ADefault when not found
4480: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4481: // same for a color
4482: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4483: // same for an Integer
4484: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4485: // same for a float
4486: function GetBoolValue(const AText, AName: string): Boolean;
4487: // same for Boolean but without default
4488: function GetValue(const AText, AName: string): string;
4489: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4490: procedure SetValue(var AText: string; const AName, AValue: string);
4491: // sets a string value in a line
4492: procedure DeleteValue(var AText: string; const AName: string);
4493: // deletes a AName="value" pair from AText
4494: procedure GetNames(AText: string; AList: TStringList);
4495: // get a list of names from a string with name="value" pairs
4496: function GetHTMLColor(AColor: TColor): string;
4497: // converts a color value to the HTML hex value
4498: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4499: // finds a string backward case sensitive
4500: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4501: // finds a string backward case insensitive
4502: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4503:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4504: // finds a text range, e.g. <TD>....</TD> case sensitive
4505: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4506:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4507: // finds a text range, e.g. <TD>....</td> case insensitive
4508: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4509:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4510: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4511: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4512:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4513: // finds a text range backward, e.g. <TD>....</td> case insensitive
4514: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4515:   var RangeEnd: Integer): Boolean;
4516: // finds a HTML or XML tag:  <....>
4517: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4518:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4519: // finds the innertext between opening and closing tags
4520: function Easter(NYear: Integer): TDateTime;
4521: // returns the easter date of a year.
4522: function GetWeekNumber(Today: TDateTime): string;
4523: //gets a datecode. Returns year and weeknumber in format: YYWW
4524: function ParseNumber(const S: string): Integer;
4525: // parse number returns the last position, starting from 1
4526: function ParseDate(const S: string): Integer;
4527: // parse a SQL style data string from positions 1,
4528: // starts and ends with #
4529:
4530: ****************************************unit JvJCLUtils;****************************************
4531:
4532: function VarIsInt(Value: Variant): Boolean;
4533:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4534: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4535: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4536: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4537: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4538: { GetWordOnPos returns Word from string, S, on the cursor position, P}
```

```
4539: function GetWordOnPos(const S: string; const P: Integer): string;
4540: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4541: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4542: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4543: { GetWordOnPosEx working like GetWordOnPos function, but
4544:   also returns Word position in iBeg, iEnd variables }
4545: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4546: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4547: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4548: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4549: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4550: { GetEndPosCaret returns the caret position of the last char. For the position
4551:   after the last char of Text you must add 1 to the returned X value. }
4552: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4553: { GetEndPosCaret returns the caret position of the last char. For the position
4554:   after the last char of Text you must add 1 to the returned X value. }
4555: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4556: function SubStrBySeparator(const S:string;const Index:Integer;const
      Separator:string;StartIndex:Int=1):string;
4557: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
      Separator:WideString;StartIndex:Int:WideString;
4558: { SubStrEnd same to previous function but Index numerated from the end of string }
4559: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4560: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4561: function SubWord(P: PChar; var P2: PChar): string;
4562: function CurrencyByWord(Value: Currency): string;
4563: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4564: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4565: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4566: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4567: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4568: { ReplaceString searches for all substrings, OldPattern,
4569:   in a string, S, and replaces them with NewPattern }
4570: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4571: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
      WideString;StartIndex:Integer=1):WideString;
4572: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4573: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4574: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4575: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4576:
4577: { Next 4 function for russian chars transliterating.
4578:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4579: procedure Dos2Win(var S: AnsiString);
4580: procedure Win2Dos(var S: AnsiString);
4581: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4582: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4583: function Win2Koi(const S: AnsiString): AnsiString;
4584: { FillString fills the string Buffer with Count Chars }
4585: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4586: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4587: { MoveString copies Count Chars from Source to Dest }
4588: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4589: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4590:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4591: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4592: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4593: { MoveWideChar copies Count WideChars from Source to Dest }
4594: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4595: { FillNativeChar fills Buffer with Count NativeChars }
4596: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4597: { MoveWideChar copies Count WideChars from Source to Dest }
4598: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4599: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4600: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4601: { Spaces returns string consists on N space chars }
4602: function Spaces(const N: Integer): string;
4603: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4604: function AddSpaces(const S: string; const N: Integer): string;
4605: function SpacesW(const N: Integer): WideString;
4606: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4607: { function LastDateRUS for russian users only }
4608: { returns date relative to current date: 'ääà äíÿ íàçàä' }
4609: function LastDateRUS(const Dat: TDateTime): string;
4610: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4611: function CurrencyToStr(const Cur: Currency): string;
4612: { HasChar returns True, if Char, Ch, contains in string, S }
4613: function HasChar(const Ch: Char; const S: string): Boolean;
4614: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4615: function HasAnyChar(const Chars: string; const S: string): Boolean;
4616: {$IFNDEF COMPILER12_UP}
4617: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4618: {$ENDIF ~COMPILER12_UP}
```

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

```
4705: function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4706: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4707: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4708: {**** Color routines }
4709: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4710: function RGBToBGR(Value: Cardinal): Cardinal;
4711: //function ColorToPrettyName(Value: TColor): string;
4712: //function PrettyNameToColor(const Value: string): TColor;
4713: {**** other routines }
4714: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4715: function IntPower(Base, Exponent: Integer): Integer;
4716: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4717: function StrToBool(const S: string): Boolean;
4718: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4719: function VarToInt(V: Variant): Integer;
4720: function VarToFloat(V: Variant): Double;
4721: { following functions are not documented because they not work properly sometimes, so do not use them }
4722: // (rom) ReplaceStrings1, GetSubStr removed
4723: function GetLongFileName(const FileName: string): string;
4724: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4725: function GetParameter: string;
4726: function GetComputerID: string;
4727: function GetComputerName: string;
4728: {**** string routines }
4729: { ReplaceAllStrings searches for all substrings, Words,
4730:   in a string, S, and replaces them with Frases with the same Index. }
4731: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4732: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4733:   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 }
4734: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4735: { CountOfLines calculates the lines count in a string, S,
4736:   each line must be separated from another with CrLf sequence }
4737: function CountOfLines(const S: string): Integer;
4738: { DeleteLines deletes all lines from strings which in the words,  words.
4739:   The word of will be deleted from strings. }
4740: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4741: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4742:   Lines contained only spaces also deletes. }
4743: procedure DeleteEmptyLines(Ss: TStrings);
4744: { SQLAddWhere addes or modifies existing where-statement, where,
4745:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4746:   it must be started on the begining of any line }
4747: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4748: {**** files routines - }
4749: {$IFDEF MSWINDOWS}
4750: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4751:   Resource can be compressed using MS Compress program}
4752: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4753: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
      Boolean;
4754: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4755: {$ENDIF MSWINDOWS}
4756: { IniReadSection read section, Section, from ini-file,
4757:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4758:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4759: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4760: { LoadTextFile load text file, FileName, into string }
4761: function LoadTextFile(const FileName: TFileName): string;
4762: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4763: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4764: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4765: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4766: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4767: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4768: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4769: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4770: { RATextCalcHeight calculate needed height for
4771:   correct output, using RATextOut or RATextOutEx functions }
4772: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4773: { Cinema draws some visual effect }
4774: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4775: { Roughed fills rect with special 3D pattern }
4776: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4777: { 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 }
4778: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4779: { TextWidth calculate text with for writing using standard desktop font }
4780: function TextWidth(const AStr: string): Integer;
4781: { TextHeight calculate text height for writing using standard desktop font }
4782: function TextHeight(const AStr: string): Integer;
4783: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4784: procedure Error(const Msg: string);
4785: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4786:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4787: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4788: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4789:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4790: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
```

```
4791:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4792: function ItemHtPlain(const Text: string): string;
4793: { ClearList - clears list of TObject }
4794: procedure ClearList(List: TList);
4795: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4796: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4797: { RTTI support }
4798: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4799: function GetPropStr(Obj: TObject; const PropName: string): string;
4800: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4801: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4802: procedure PrepareIniSection(Ss: TStrings);
4803: { following functions are not documented because they are don't work properly, so don't use them }
4804: // (rom) from JvBandWindows to make it obsolete
4805: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4806: // (rom) from JvBandUtils to make it obsolete
4807: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4808: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4809: function CreateIconFromClipboard: TIcon;
4810: { begin JvIconClipboardUtils } { Icon clipboard routines }
4811: function CF_ICON: Word;
4812: procedure AssignClipboardIcon(Icon: TIcon);
4813: { Real-size icons support routines (32-bit only) }
4814: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4815: function CreateRealSizeIcon(Icon: TIcon): HICON;
4816: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4817: {end JvIconClipboardUtils }
4818: function CreateScreenCompatibleDC: HDC;
4819: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF}
4820: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4821: { begin JvRLE } // (rom) changed API for inclusion in JCL
4822: procedure RleCompressTo(InStream, OutStream: TStream);
4823: procedure RleDecompressTo(InStream, OutStream: TStream);
4824: procedure RleCompress(Stream: TStream);
4825: procedure RleDecompress(Stream: TStream);
4826: { end JvRLE } { begin JvDateUtil }
4827: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4828: function IsLeapYear(AYear: Integer): Boolean;
4829: function DaysInAMonth(const AYear, AMonth: Word): Word;
4830: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4831: function FirstDayOfPrevMonth: TDateTime;
4832: function LastDayOfPrevMonth: TDateTime;
4833: function FirstDayOfNextMonth: TDateTime;
4834: function ExtractDay(ADate: TDateTime): Word;
4835: function ExtractMonth(ADate: TDateTime): Word;
4836: function ExtractYear(ADate: TDateTime): Word;
4837: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4838: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4839: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4840: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4841: function ValidDate(ADate: TDateTime): Boolean;
4842: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4843: function MonthsBetween(Date1, Date2: TDateTime): Double;
4844: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4845: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4846: function DaysBetween(Date1, Date2: TDateTime): Longint;
4847: { The same as previous but if Date2 < Date1 result = 0 }
4848: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4849: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4850: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4851: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4852: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4853: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4854: { String to date conversions }
4855: function GetDateOrder(const DateFormat: string): TDateOrder;
4856: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4857: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4858: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4859: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4860: //function DefDateFormat(AFourDigitYear: Boolean): string;
4861: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4862: function FormatLongDate(Value: TDateTime): string;
4863: function FormatLongDateTime(Value: TDateTime): string;
4864: { end JvDateUtil }
4865: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4866: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4867: { begin JvStrUtils } { ** Common string handling routines ** }
4868: {$IFDEF UNIX}
4869: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4870:   const ToCode, FromCode: AnsiString): Boolean;
4871: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4872: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4873: function OemStrToAnsi(const S: AnsiString): AnsiString;
4874: function AnsiStrToOem(const S: AnsiString): AnsiString;
4875: {$ENDIF UNIX}
4876: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4877: { StrToOem translates a string from the Windows character set into the OEM character set. }
4878: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
```

```
4879: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4880: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4881: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4882: function ReplaceStr(const S, Srch, Replace: string): string;
4883: { Returns string with every occurrence of Srch string replaced with Replace string. }
4884: function DelSpace(const S: string): string;
4885: { DelSpace return a string with all white spaces removed. }
4886: function DelChars(const S: string; Chr: Char): string;
4887: { DelChars return a string with all Chr characters removed. }
4888: function DelBSpace(const S: string): string;
4889: { DelBSpace trims leading spaces from the given string. }
4890: function DelESpace(const S: string): string;
4891: { DelESpace trims trailing spaces from the given string. }
4892: function DelRSpace(const S: string): string;
4893: { DelRSpace trims leading and trailing spaces from the given string. }
4894: function DelSpace1(const S: string): string;
4895: { DelSpace1 return a string with all non-single white spaces removed. }
4896: function Tab2Space(const S: string; Numb: Byte): string;
4897: { Tab2Space converts any tabulation character in the given string to the
4898:   Numb spaces characters. }
4899: function NPos(const C: string; S: string; N: Integer): Integer;
4900: { NPos searches for a N-th position of substring C in a given string. }
4901: function MakeStr(C: Char; N: Integer): string; overload;
4902: {$IFNDEF COMPILER12_UP}
4903: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4904: {$ENDIF !COMPILER12_UP}
4905: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4906: { MakeStr return a string of length N filled with character C. }
4907: function AddChar(C: Char; const S: string; N: Integer): string;
4908: { AddChar return a string left-padded to length N with characters C. }
4909: function AddCharR(C: Char; const S: string; N: Integer): string;
4910: { AddCharR return a string right-padded to length N with characters C. }
4911: function LeftStr(const S: string; N: Integer): string;
4912: { LeftStr return a string right-padded to length N with blanks. }
4913: function RightStr(const S: string; N: Integer): string;
4914: { RightStr return a string left-padded to length N with blanks. }
4915: function CenterStr(const S: string; Len: Integer): string;
4916: { CenterStr centers the characters in the string based upon the Len specified. }
4917: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4918: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4919:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4920: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4921: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4922: function Copy2Symb(const S: string; Symb: Char): string;
4923: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4924: function Copy2SymbDel(var S: string; Symb: Char): string;
4925: { Copy2SymbDel returns a substring of a string S from begining to first
4926:   character Symb and removes this substring from S. }
4927: function Copy2Space(const S: string): string;
4928: { Copy2Symb returns a substring of a string S from begining to first white space. }
4929: function Copy2SpaceDel(var S: string): string;
4930: { Copy2SpaceDel returns a substring of a string S from begining to first
4931:   white space and removes this substring from S. }
4932: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4933: { Returns string, with the first letter of each word in uppercase,
4934:   all other letters in lowercase. Words are delimited by WordDelims. }
4935: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4936: { WordCount given a set of word delimiters, returns number of words in S. }
4937: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4938: { Given a set of word delimiters, returns start position of N'th word in S. }
4939: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4940: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4941: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4942: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4943:   delimiters, return the N'th word in S. }
4944: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4945: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4946:   that started from position Pos. }
4947: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4948: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4949: function QuotedString(const S: string; Quote: Char): string;
4950: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4951: function ExtractQuotedString(const S: string; Quote: Char): string;
4952: { ExtractQuotedString removes the Quote characters from the beginning and
4953:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4954: function FindPart(const HelpWilds, InputStr: string): Integer;
4955: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4956: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4957: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4958: function XorString(const Key, Src: ShortString): ShortString;
4959: function XorEncode(const Key, Source: string): string;
4960: function XorDecode(const Key, Source: string): string;
4961: { ** Command line routines ** }
4962: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4963: { ** Numeric string handling routines ** }
4964: function Numb2USA(const S: string): string;
4965: { Numb2USA converts numeric string S to USA-format. }
4966: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4967: { Dec2Hex converts the given value to a hexadecimal string representation
```

```
4968:   with the minimum number of digits (A) specified. }
4969: function Hex2Dec(const S: string): Longint;
4970: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4971: function Dec2Numb(N: Int64; A, B: Byte): string;
4972: { Dec2Numb converts the given value to a string representation with the
4973:   base equal to B and with the minimum number of digits (A) specified. }
4974: function Numb2Dec(S: string; B: Byte): Int64;
4975: { Numb2Dec converts the given B-based numeric string to the corresponding
4976:   integer value. }
4977: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4978: { IntToBin converts the given value to a binary string representation
4979:   with the minimum number of digits specified. }
4980: function IntToRoman(Value: Longint): string;
4981: { IntToRoman converts the given value to a roman numeric string representation. }
4982: function RomanToInt(const S: string): Longint;
4983: { RomanToInt converts the given string to an integer value. If the string
4984:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4985: function FindNotBlankCharPos(const S: string): Integer;
4986: function FindNotBlankCharPosW(const S: WideString): Integer;
4987: function AnsiChangeCase(const S: string): string;
4988: function WideChangeCase(const S: string): string;
4989: function StartsText(const SubStr, S: string): Boolean;
4990: function EndsText(const SubStr, S: string): Boolean;
4991: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4992: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4993: {end JvStrUtils}
4994: {$IFDEF UNIX}
4995: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4996: {$ENDIF UNIX}
4997: { begin JvFileUtil }
4998: function FileDateTime(const FileName: string): TDateTime;
4999: function HasAttr(const FileName: string; Attr: Integer): Boolean;
5000: function DeleteFilesEx(const FileMasks: array of string): Boolean;
5001: function NormalDir(const DirName: string): string;
5002: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
5003: function ValidFileName(const FileName: string): Boolean;
5004: {$IFDEF MSWINDOWS}
5005: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5006: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5007: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5008: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5009: {$ENDIF MSWINDOWS}
5010: function GetWindowsDir: string;
5011: function GetSystemDir: string;
5012: function ShortToLongFileName(const ShortName: string): string;
5013: function LongToShortFileName(const LongName: string): string;
5014: function ShortToLongPath(const ShortName: string): string;
5015: function LongToShortPath(const LongName: string): string;
5016: {$IFDEF MSWINDOWS}
5017: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
5018: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5019: {$ENDIF MSWINDOWS}
5020: { end JvFileUtil }
5021: // Works like PtInRect but includes all edges in comparision
5022: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5023: // Works like PtInRect but excludes all edges from comparision
5024: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5025: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5026: function IsFourDigitYear: Boolean;
5027: { moved from JvJVCLUTils }
5028: //Open an object with the shell (url or something like that)
5029: function OpenObject(const Value: string): Boolean; overload;
5030: function OpenObject(Value: PChar): Boolean; overload;
5031: {$IFDEF MSWINDOWS}
5032: //Raise the last Exception
5033: procedure RaiseLastWin32; overload;
5034: procedure RaiseLastWin32(const Text: string); overload;
5035: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
5036: function GetFileVersion(const AFileName: string): Cardinal;
5037: {$EXTERNALSYM GetFileVersion}
5038: //Get version of Shell.dll
5039: function GetShellVersion: Cardinal;
5040: {$EXTERNALSYM GetShellVersion}
5041: // CD functions  on HW
5042: procedure OpenCdDrive;
5043: procedure CloseCdDrive;
5044: // returns True if Drive is accessible
5045: function DiskInDrive(Drive: Char): Boolean;
5046: {$ENDIF MSWINDOWS}
5047: //Same as linux function ;)
5048: procedure PError(const Text: string);
5049: // execute a program without waiting
5050: procedure Exec(const FileName, Parameters, Directory: string);
5051: // execute a program and wait for it to finish
5052: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5053: // returns True if this is the first instance of the program that is running
5054: function FirstInstance(const ATitle: string): Boolean;
```

```
5055: // restores a window based on it's classname and Caption. Either can be left empty
5056: // to widen the search
5057: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5058: // manipulate the traybar and start button
5059: procedure HideTraybar;
5060: procedure ShowTraybar;
5061: procedure ShowStartButton(Visible: Boolean = True);
5062: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5063: procedure MonitorOn;
5064: procedure MonitorOff;
5065: procedure LowPower;
5066: // send a key to the window named AppName
5067: function SendKey(const AppName: string; Key: Char): Boolean;
5068: {$IFDEF MSWINDOWS}
5069: // returns a list of all win currently visible, the Objects property is filled with their window handle
5070: procedure GetVisibleWindows(List: TStrings);
5071: Function GetVisibleWindowsF( List : TStrings):TStrings');
5072: // associates an extension to a specific program
5073: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5074: procedure AddToRecentDocs(const FileName: string);
5075: function GetRecentDocs: TStringList;
5076: {$ENDIF MSWINDOWS}
5077: function CharIsMoney(const Ch: Char): Boolean;
5078: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5079: function IntToExtended(I: Integer): Extended;
5080: { GetChangedText works out the new text given the current cursor pos & the key pressed
5081:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5082: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5083: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5084: //function StrIsInteger(const S: string): Boolean;
5085: function StrIsFloatMoney(const Ps: string): Boolean;
5086: function StrIsDateTime(const Ps: string): Boolean;
5087: function PreformatDateString(Ps: string): string;
5088: function BooleanToInteger(const B: Boolean): Integer;
5089: function StringToBoolean(const Ps: string): Boolean;
5090: function SafeStrToDateTime(const Ps: string): TDateTime;
5091: function SafeStrToDate(const Ps: string): TDateTime;
5092: function SafeStrToTime(const Ps: string): TDateTime;
5093: function StrDelete(const psSub, psMain: string): string;
5094:   { returns the fractional value of pcValue}
5095: function TimeOnly(pcValue: TDateTime): TTime;
5096: { returns the integral value of pcValue }
5097: function DateOnly(pcValue: TDateTime): TDate;
5098: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5099: const { TDateTime value used to signify Null value}
5100:   NullEquivalentDate: TDateTime = 0.0;
5101: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5102: // Replacement for Win32Check to avoid platform specific warnings in D6
5103: function OSCheck(RetVal: Boolean): Boolean;
5104: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5105:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5106:   not be forced to use FileCtrl unnecessarily }
5107: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5108: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5109: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5110: function MinimizeString(const S: string; const MaxLen: Integer): string;
5111: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
5112: { 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.}
5113: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5114: {$ENDIF MSWINDOWS}
5115: procedure ResourceNotFound(ResID: PChar);
5116: function EmptyRect: TRect;
5117: function RectWidth(R: TRect): Integer;
5118: function RectHeight(R: TRect): Integer;
5119: function CompareRect(const R1, R2: TRect): Boolean;
5120: procedure RectNormalize(var R: TRect);
5121: function RectIsSquare(const R: TRect): Boolean;
5122: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5123: //If AMaxSize = -1 ,then auto calc Square's max size
5124: {$IFDEF MSWINDOWS}
5125: procedure FreeUnusedOle;
5126: function GetWindowsVersion: string;
5127: function LoadDLL(const LibName: string): THandle;
5128: function RegisterServer(const ModuleName: string): Boolean;
5129: function UnregisterServer(const ModuleName: string): Boolean;
5130: {$ENDIF MSWINDOWS}
5131: { String routines }
5132: function GetEnvVar(const VarName: string): string;
5133: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5134: function StringToPChar(var S: string): PChar;
5135: function StrPAlloc(const S: string): PChar;
5136: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5137: function DropT(const S: string): string;
5138: { Memory routines }
5139: function AllocMemo(Size: Longint): Pointer;
5140: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
```

```
5141: procedure FreeMemo(var fpBlock: Pointer);
5142: function GetMemoSize(fpBlock: Pointer): Longint;
5143: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5144: { Manipulate huge pointers routines }
5145: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5146: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5147: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5148: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5149: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5150: function WindowClassName(Wnd: THandle): string;
5151: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5152: procedure ActivateWindow(Wnd: THandle);
5153: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5154: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5155: { SetWindowTop put window to top without recreating window }
5156: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5157: procedure CenterWindow(Wnd: THandle);
5158: function MakeVariant(const Values: array of Variant): Variant;
5159: { Convert dialog units to pixels and backwards }
5160: {$IFDEF MSWINDOWS}
5161: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5162: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5163: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5164: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5165: {$ENDIF MSWINDOWS}
5166: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5167: {$IFDEF BCB}
5168: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5169: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5170: {$ELSE}
5171: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5172: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5173: {$ENDIF BCB}
5174: {$IFDEF MSWINDOWS}
5175: { BrowseForFolderNative displays Browse For Folder dialog }
5176: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5177: {$ENDIF MSWINDOWS}
5178: procedure AntiAlias(Clip: TBitmap);
5179: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5180: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5181:   ABitmap: TBitmap; const SourceRect: TRect);
5182: function IsTrueType(const FontName: string): Boolean;
5183: // Removes all non-numeric characters from AValue and returns the resulting string
5184: function TextToValText(const AValue: string): string;
5185: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5186: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5187: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5188: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5189: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5190:
5191: ******************************************************unit uPSI_JvTFUtils;
5192:  Function JExtractYear( ADate : TDateTime) : Word
5193:  Function JExtractMonth( ADate : TDateTime) : Word
5194:  Function JExtractDay( ADate : TDateTime) : Word
5195:  Function ExtractHours( ATime : TDateTime) : Word
5196:  Function ExtractMins( ATime : TDateTime) : Word
5197:  Function ExtractSecs( ATime : TDateTime) : Word
5198:  Function ExtractMSecs( ATime : TDateTime) : Word
5199:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5200:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5201:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5202:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5203:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5204:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5205:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5206:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5207:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5208:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5209:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5210:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5211:  Procedure EnsureMonth( Month : Word)
5212:  Procedure EnsureDOW( DOW : Word)
5213:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5214:  Function Lesser( N1, N2 : Integer) : Integer
5215:  Function Greater( N1, N2 : Integer) : Integer
5216:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5217:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5218:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5219:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5220:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5221:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5222:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
     AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5223:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
     HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5224:  Function JRectWidth( ARect : TRect) : Integer
5225:  Function JRectHeight( ARect : TRect) : Integer
5226:  Function JEmptyRect : TRect
```

```
5227:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5228:
5229: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5230: begin
5231:  Procedure HideTaskBarButton( hWindow : HWND)
5232:  Function msLoadStr( ID : Integer) : String
5233:  Function msFormat( fmt : String; params : array of const) : String
5234:  Function msFileExists( const FileName : String) : Boolean
5235:  Function msIntToStr( Int : Int64) : String
5236:  Function msStrPas( const Str : PChar) : String
5237:  Function msRenameFile( const OldName, NewName : String) : Boolean
5238:  Function CutFileName( s : String) : String
5239:  Function GetVersionInfo( var VersionString : String) : DWORD
5240:  Function FormatTime( t : Cardinal) : String
5241:  Function msCreateDir( const Dir : string) : Boolean
5242:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5243:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5244:  Function msStrLen( Str : PChar) : Integer
5245:  Function msDirectoryExists( const Directory : String) : Boolean
5246:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5247:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5248:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5249:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5250:  Function GetTextFromFile( Filename : String) : string
5251:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5252:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5253:  Function msStrToInt( s : String) : Integer
5254:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5255: end;
5256:
5257: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5258: begin
5259:   //TDynFloatArray', 'array of Extended
5260:   TDynLWordArray', 'array of LongWord
5261:   TDynLIntArray', 'array of LongInt
5262:   TDynFloatMatrix', 'array of TDynFloatArray
5263:   TDynLWordMatrix', 'array of TDynLWordArray
5264:   TDynLIntMatrix', 'array of TDynLIntArray
5265:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5266:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5267:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5268:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5269:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5270:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5271:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5272:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5273:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5274:  Function MNorm( const X : TDynFloatArray) : Extended
5275:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5276:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5277:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5278:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5279:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5280:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5281:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5282:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5283:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5284:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5285:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5286:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5287:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5288: end;
5289:
5290: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5291: begin
5292:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5293:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5294:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5295:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5296:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5297:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5298:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5299:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5300:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5301:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5302:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5303:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5304:  'ESBSqrtPi','Extended').setExtended( 1.7724538509055160272729);
5305:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5306:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5307:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5308:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5309:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5310:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5311:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5312:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5313:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5314:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5315:  'ESBe','Extended').setExtended( 2.7182818284590452354);
```

```
5316:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5317:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5318:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5319:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5320:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5321:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5322:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5323:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5324:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5325:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5326:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5327:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5328:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5329:  'ESBInvPi','Extended').setExtended( 3.18309886183790671 54e-1);
5330:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5331:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5332:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5333:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5334:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5335:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5336:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5337:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5338:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5339:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5340:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5341:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5342:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5343:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5344:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5345:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5346:   //LongWord', 'Cardinal
5347:   TBitList', 'Word
5348:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5349:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5350:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5351:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5352:  Function SameFloat( const X1, X2 : Extended) : Boolean
5353:  Function FloatIsZero( const X : Extended) : Boolean
5354:  Function FloatIsPositive( const X : Extended) : Boolean
5355:  Function FloatIsNegative( const X : Extended) : Boolean
5356:  Procedure IncLim( var B : Byte; const Limit : Byte)
5357:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5358:  Procedure IncLimW( var B : Word; const Limit : Word)
5359:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5360:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5361:  Procedure DecLim( var B : Byte; const Limit : Byte)
5362:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5363:  Procedure DecLimW( var B : Word; const Limit : Word)
5364:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5365:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5366:  Function MaxB( const B1, B2 : Byte) : Byte
5367:  Function MinB( const B1, B2 : Byte) : Byte
5368:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5369:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5370:  Function MaxW( const B1, B2 : Word) : Word
5371:  Function MinW( const B1, B2 : Word) : Word
5372:  Function esbMaxI( const B1, B2 : Integer) : Integer
5373:  Function esbMinI( const B1, B2 : Integer) : Integer
5374:  Function MaxL( const B1, B2 : LongInt) : LongInt
5375:  Function MinL( const B1, B2 : LongInt) : LongInt
5376:  Procedure SwapB( var B1, B2 : Byte)
5377:  Procedure SwapSI( var B1, B2 : ShortInt)
5378:  Procedure SwapW( var B1, B2 : Word)
5379:  Procedure SwapI( var B1, B2 : SmallInt)
5380:  Procedure SwapL( var B1, B2 : LongInt)
5381:  Procedure SwapI32( var B1, B2 : Integer)
5382:  Procedure SwapC( var B1, B2 : LongWord)
5383:  Procedure SwapInt64( var X, Y : Int64)
5384:  Function esbSign( const B : LongInt) : ShortInt
5385:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5386:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5387:  Function Max3Word( const X1, X2, X3 : Word) : Word
5388:  Function Min3Word( const X1, X2, X3 : Word) : Word
5389:  Function MaxBArray( const B : array of Byte) : Byte
5390:  Function MaxWArray( const B : array of Word) : Word
5391:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5392:  Function MaxIArray( const B : array of Integer) : Integer
5393:  Function MaxLArray( const B : array of LongInt) : LongInt
5394:  Function MinBArray( const B : array of Byte) : Byte
5395:  Function MinWArray( const B : array of Word) : Word
5396:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5397:  Function MinIArray( const B : array of Integer) : Integer
5398:  Function MinLArray( const B : array of LongInt) : LongInt
5399:  Function SumBArray( const B : array of Byte) : Byte
5400:  Function SumBArray2( const B : array of Byte) : Word
5401:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5402:  Function SumSIArray2( const B : array of ShortInt) : Integer
5403:  Function SumWArray( const B : array of Word) : Word
5404:  Function SumWArray2( const B : array of Word) : LongInt
```

```
5405:  Function SumIArray( const B : array of Integer) : Integer
5406:  Function SumLArray( const B : array of LongInt) : LongInt
5407:  Function SumLWArray( const B : array of LongWord) : LongWord
5408:  Function ESBDigits( const X : LongWord) : Byte
5409:  Function BitsHighest( const X : LongWord) : Integer
5410:  Function ESBBitsNeeded( const X : LongWord) : Integer
5411:  Function esbGCD( const X, Y : LongWord) : LongWord
5412:  Function esbLCM( const X, Y : LongInt) : Int64
5413:  //Function esbLCM( const X, Y : LongInt) : LongInt
5414:  Function RelativePrime( const X, Y : LongWord) : Boolean
5415:  Function Get87ControlWord : TBitList
5416:  Procedure Set87ControlWord( const CWord : TBitList)
5417:  Procedure SwapExt( var X, Y : Extended)
5418:  Procedure SwapDbl( var X, Y : Double)
5419:  Procedure SwapSing( var X, Y : Single)
5420:  Function esbSgn( const X : Extended) : ShortInt
5421:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5422:  Function ExtMod( const X, Y : Extended) : Extended
5423:  Function ExtRem( const X, Y : Extended) : Extended
5424:  Function CompMOD( const X, Y : Comp) : Comp
5425:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5426:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5427:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5428:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5429:  Function MaxExt( const X, Y : Extended) : Extended
5430:  Function MinExt( const X, Y : Extended) : Extended
5431:  Function MaxEArray( const B : array of Extended) : Extended
5432:  Function MinEArray( const B : array of Extended) : Extended
5433:  Function MaxSArray( const B : array of Single) : Single
5434:  Function MinSArray( const B : array of Single) : Single
5435:  Function MaxCArray( const B : array of Comp) : Comp
5436:  Function MinCArray( const B : array of Comp) : Comp
5437:  Function SumSArray( const B : array of Single) : Single
5438:  Function SumEArray( const B : array of Extended) : Extended
5439:  Function SumSqEArray( const B : array of Extended) : Extended
5440:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5441:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5442:  Function SumCArray( const B : array of Comp) : Comp
5443:  Function FactorialX( A : LongWord) : Extended
5444:  Function PermutationX( N, R : LongWord) : Extended
5445:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5446:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5447:  Function esbGeometricMean( const X : array of Extended) : Extended
5448:  Function esbHarmonicMean( const X : array of Extended) : Extended
5449:  Function ESBMean( const X : array of Extended) : Extended
5450:  Function esbSampleVariance( const X : array of Extended) : Extended
5451:  Function esbPopulationVariance( const X : array of Extended) : Extended
5452:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5453:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5454:  Function GetMedian( const SortedX : array of Extended) : Extended
5455:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5456:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5457:  Function ESBMagnitude( const X : Extended) : Integer
5458:  Function ESBTan( Angle : Extended) : Extended
5459:  Function ESBCot( Angle : Extended) : Extended
5460:  Function ESBCosec( const Angle : Extended) : Extended
5461:  Function ESBSec( const Angle : Extended) : Extended
5462:  Function ESBArcTan( X, Y : Extended) : Extended
5463:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5464:  Function ESBArcCos( const X : Extended) : Extended
5465:  Function ESBArcSin( const X : Extended) : Extended
5466:  Function ESBArcSec( const X : Extended) : Extended
5467:  Function ESBArcCosec( const X : Extended) : Extended
5468:  Function ESBLog10( const X : Extended) : Extended
5469:  Function ESBLog2( const X : Extended) : Extended
5470:  Function ESBLogBase( const X, Base : Extended) : Extended
5471:  Function Pow2( const X : Extended) : Extended
5472:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5473:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5474:  Function XtoY( const X, Y : Extended) : Extended
5475:  Function esbTenToY( const Y : Extended) : Extended
5476:  Function esbTwoToY( const Y : Extended) : Extended
5477:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5478:  Function esbISqrt( const I : LongWord) : Longword
5479:  Function ILog2( const I : LongWord) : LongWord
5480:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5481:  Function ESBArCosh( X : Extended) : Extended
5482:  Function ESBArSinh( X : Extended) : Extended
5483:  Function ESBArTanh( X : Extended) : Extended
5484:  Function ESBCosh( X : Extended) : Extended
5485:  Function ESBSinh( X : Extended) : Extended
5486:  Function ESBTanh( X : Extended) : Extended
5487:  Function InverseGamma( const X : Extended) : Extended
5488:  Function esbGamma( const X : Extended) : Extended
5489:  Function esbLnGamma( const X : Extended) : Extended
5490:  Function esbBeta( const X, Y : Extended) : Extended
5491:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5492: end;
5493:
```

```
5494: ******************************Integer Huge Cardinal Utils*************************
5495: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5496: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5497: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5498: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5499: Function BitCount_8( Value : byte) : integer
5500: Function BitCount_16( Value : uint16) : integer
5501: Function BitCount_32( Value : uint32) : integer
5502: Function BitCount_64( Value : uint64) : integer
5503: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5504:  Procedure ( CountPrimalityTests : integer)
5505:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5506:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5507:  Function isCoPrime( a, b : THugeCardinal) : boolean
5508:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5509:  Function hasSmallFactor( p : THugeCardinal) : boolean
5510:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5511:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5512:  Const('StandardExponent','LongInt'( 65537);
5513: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5514:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5515:
5516: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5517: begin
5518:   AddTypeS('TXRTLInteger', 'array of Integer
5519:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5520:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5521:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5522:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5523:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5524:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5525:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5526:   'BitsPerByte','LongInt'( 8);
5527:   BitsPerDigit','LongInt'( 32);
5528:   SignBitMask','LongWord( $80000000);
5529:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5530:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5531:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5532:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5533:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5534:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5535:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5536:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5537:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5538:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5539:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5540:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5541:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5542:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5543:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5544:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5545:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5546:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5547:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5548:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5549:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5550:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5551:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5552:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5553:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5554:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5555:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5556:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5557:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5558:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5559:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5560:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5561:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5562:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5563:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5564:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5565:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5566:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5567:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5568:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5569:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5570:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5571:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5572:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5573:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5574:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5575:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5576:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
```

```
5577:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5578:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5579:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5580:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5581:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5582:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5583:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5584:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5585:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5586:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5587:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5588:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5589:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5590:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5591:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5592:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5593:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5594:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5595:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5596:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5597:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5598:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5599:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5600:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5601:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5602:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5603: end;
5604:
5605: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5606: begin
5607:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5608:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5609:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5610:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5611:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5612:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5613:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5614:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5615:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5616:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5617:  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)
5618: end;
5619:
5620:
5621: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5622: begin
5623:  Function StrDec( S : String) : String
5624:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5625:  Function ReadNumFromEdit( Edit : TEdit) : Float
5626:  Procedure WriteNumToFile( var F : Text; X : Float)
5627: end;
5628:
5629: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5630: begin
5631:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5632:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5633:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5634:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5635:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5636:  Procedure TeX_SetGraphTitle( Title : String)
5637:  Procedure TeX_SetOxTitle( Title : String)
5638:  Procedure TeX_SetOyTitle( Title : String)
5639:  Procedure TeX_PlotOxAxis
5640:  Procedure TeX_PlotOyAxis
5641:  Procedure TeX_PlotGrid( Grid : TGrid)
5642:  Procedure TeX_WriteGraphTitle
5643:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5644:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5645:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5646:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5647:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5648:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5649:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5650:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5651:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5652:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5653:  Function Xcm( X : Float) : Float
5654:  Function Ycm( Y : Float) : Float
5655: end;
5656:
5657: *---------------------------------------------------------------------------*)
5658: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
```

```
5659: begin
5660:   TConstArray', 'array of TVarRec
5661:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5662:  Function CreateConstArray( const Elements : array of const) : TConstArray
5663:  Procedure FinalizeVarRec( var Item : TVarRec)
5664:  Procedure FinalizeConstArray( var Arr : TConstArray)
5665: end;
5666:
5667: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5668: begin
5669:  Function HexBS( B : Byte) : ShortString
5670:  Function HexWS( W : Word) : ShortString
5671:  Function HexLS( L : LongInt) : ShortString
5672:  Function HexPtrS( P : Pointer) : ShortString
5673:  Function BinaryBS( B : Byte) : ShortString
5674:  Function BinaryWS( W : Word) : ShortString
5675:  Function BinaryLS( L : LongInt) : ShortString
5676:  Function OctalBS( B : Byte) : ShortString
5677:  Function OctalWS( W : Word) : ShortString
5678:  Function OctalLS( L : LongInt) : ShortString
5679:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5680:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5681:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5682:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5683:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5684:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5685:  Function Long2StrS( L : LongInt) : ShortString
5686:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5687:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5688:  Function ValPrepS( const S : ShortString) : ShortString
5689:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5690:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5691:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5692:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5693:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5694:  Function TrimLeadS( const S : ShortString) : ShortString
5695:  Function TrimTrailS( const S : ShortString) : ShortString
5696:  Function TrimS( const S : ShortString) : ShortString
5697:  Function TrimSpacesS( const S : ShortString) : ShortString
5698:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5699:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5700:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5701:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5702:  Function ScrambleS( const S, Key : ShortString) : ShortString
5703:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5704:  Function FilterS( const S, Filters : ShortString) : ShortString
5705:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5706:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5707:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5708:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5709:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5710:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5711:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5712:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5713:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5714:  Function CompStringS( const S1, S2 : ShortString) : Integer
5715:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5716:  Function SoundexS( const S : ShortString) : ShortString
5717:  Function MakeLetterSetS( const S : ShortString) : Longint
5718:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5719:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5720:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
      Pos:Cardinal):Bool;
5721:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5722:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5723:  Function JustFilenameS( const PathName : ShortString) : ShortString
5724:  Function JustNameS( const PathName : ShortString) : ShortString
5725:  Function JustExtensionS( const Name : ShortString) : ShortString
5726:  Function JustPathnameS( const PathName : ShortString) : ShortString
5727:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5728:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5729:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5730:  Function CommaizeS( L : LongInt) : ShortString
5731:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5732:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
      DecPt:Char):ShortString;
5733:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5734:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5735:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5736:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5737:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5738:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5739:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5740:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5741:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5742:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
```

```
5743:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5744:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5745:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5746:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5747:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
5748:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
      SubStr:ShortString):Bool;
5749:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5750:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5751:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5752:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5753:  Function ExtractTokensS(const S,
      Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5754:  Function IsChAlphaS( C : Char) : Boolean
5755:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5756:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5757:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5758:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5759:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5760:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5761:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5762:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5763:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5764:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5765:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
      Cardinal):ShortString;
5766:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
      Replacements:Cardinal):ShortString;
5767:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5768:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
      Replacements:Cardinal):ShortString
5769:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
      Replacements:Cardinal):ShortString
5770:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5771:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5772:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5773:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5774: end;
5775:
5776:
5777: ********unit uPSI_StUtils; from Systools4************************************************************
5778: Function SignL( L : LongInt) : Integer
5779: Function SignF( F : Extended) : Integer
5780: Function MinWord( A, B : Word) : Word
5781: Function MidWord( W1, W2, W3 : Word) : Word
5782: Function MaxWord( A, B : Word) : Word
5783: Function MinLong( A, B : LongInt) : LongInt
5784: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5785: Function MaxLong( A, B : LongInt) : LongInt
5786: Function MinFloat( F1, F2 : Extended) : Extended
5787: Function MidFloat( F1, F2, F3 : Extended) : Extended
5788: Function MaxFloat( F1, F2 : Extended) : Extended
5789: Function MakeInteger16( H, L : Byte) : SmallInt
5790: Function MakeWordS( H, L : Byte) : Word
5791: Function SwapNibble( B : Byte) : Byte
5792: Function SwapWord( L : LongInt) : LongInt
5793: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5794: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5795: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5796: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5797: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5798: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5799: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5800: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5801: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5802: Procedure ExchangeBytes( var I, J : Byte)
5803: Procedure ExchangeWords( var I, J : Word)
5804: Procedure ExchangeLongInts( var I, J : LongInt)
5805: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5806: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5807: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5808: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5809: //*****************uPSI_StFIN;************************************************************
5810: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5811: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
      Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5812: Function BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5813: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
      Extended
5814: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5815: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5816: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5817: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
```

```
5818: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5819: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5820: Function DollarToDecimalText( DecDollar : Extended) : string
5821: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5822: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5823: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5824: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5825: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5826: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5827: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5828: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5829: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5830: Function IsCardValid( const S : string) : Boolean
5831: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5832: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5833: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5834: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5835: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5836: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5837: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5838: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5839: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5840: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5841: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5842: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5843: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5844: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5845: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5846: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5847: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5848: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5849: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5850: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5851:
5852: //*****************************************unit uPSI_StAstroP;
5853: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5854: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****************************************
5855: Function AveDev( const Data : array of Double) : Double
5856: Function AveDev16( const Data, NData : Integer) : Double
5857: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5858: Function Correlation( const Data1, Data2 : array of Double) : Double
5859: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5860: Function Covariance( const Data1, Data2 : array of Double) : Double
5861: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5862: Function DevSq( const Data : array of Double) : Double
5863: Function DevSq16( const Data, NData : Integer) : Double
5864: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5865:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5866: Function GeometricMeanS( const Data : array of Double) : Double
5867: Function GeometricMean16( const Data, NData : Integer) : Double
5868: Function HarmonicMeanS( const Data : array of Double) : Double
5869: Function HarmonicMean16( const Data, NData : Integer) : Double
5870: Function Largest( const Data : array of Double; K : Integer) : Double
5871: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5872: Function MedianS( const Data : array of Double) : Double
5873: Function Median16( const Data, NData : Integer) : Double
5874: Function Mode( const Data : array of Double) : Double
5875: Function Mode16( const Data, NData : Integer) : Double
5876: Function Percentile( const Data : array of Double; K : Double) : Double
5877: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5878: Function PercentRank( const Data : array of Double; X : Double) : Double
5879: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5880: Function Permutations( Number, NumberChosen : Integer) : Extended
5881: Function Combinations( Number, NumberChosen : Integer) : Extended
5882: Function FactorialS( N : Integer) : Extended
5883: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5884: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5885: Function Smallest( const Data : array of Double; K : Integer) : Double
5886: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5887: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5888: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5889:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5890:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5891: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5892: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5893: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5894: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5895: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5896: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5897: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
```

```
5898: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5899: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5900: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5901: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5902: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5903: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5904: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5905: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5906: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5907: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5908: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5909: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5910: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5911: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5912: Function NormSDist( Z : Single) : Single
5913: Function NormSInv( Probability : Single) : Single
5914: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5915: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5916: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5917: Function Erfc( X : Single) : Single
5918: Function GammaLn( X : Single) : Single
5919: Function LargestSort( const Data : array of Double; K : Integer) : Double
5920: Function SmallestSort( const Data : array of double; K : Integer) : Double
5921:
5922: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5923:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5924:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5925:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5926:  Function DefaultMergeName( MergeNum : Integer) : string
5927:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5928:
5929: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5930: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5931:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5932:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5933:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5934:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5935:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5936:  Function LunarPhase( UT : TStDateTimeRec) : Double
5937:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5938:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5939:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5940:  Function FullMoon( D : TStDate) : TStLunarRecord
5941:  Function LastQuarter( D : TStDate) : TStLunarRecord
5942:  Function NewMoon( D : TStDate) : TStLunarRecord
5943:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5944:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5945:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5946:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5947:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5948:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5949:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5950:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5951:  Function SiderealTime( UT : TStDateTimeRec) : Double
5952:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5953:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5954:  Function SEaster( Y, Epoch : Integer) : TStDate
5955:  Function DateTimeToAJD( D : TDateTime) : Double
5956:  Function HoursMin( RA : Double) : ShortString
5957:  Function DegsMin( DC : Double) : ShortString
5958:  Function AJDToDateTime( D : Double) : TDateTime
5959:
5960: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5961:  Function CurrentDate : TStDate
5962:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5963:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5964:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5965:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5966:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5967:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5968:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5969:  Function WeekOfYear( Julian : TStDate) : Byte
5970:  Function AstJulianDate( Julian : TStDate) : Double
5971:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5972:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5973:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5974:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5975:  Function StIsLeapYear( Year : Integer) : Boolean
5976:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5977:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5978:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5979:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5980:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5981:  Function CurrentTime : TStTime
5982:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5983:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5984:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5985:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5986:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
```

```
5987:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5988:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5989:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5990:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5991:  Function StDateToDateTime( D : TStDate) : TDateTime
5992:  Function StTimeToDateTime( T : TStTime) : TDateTime
5993:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5994:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5995:
5996: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5997: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5998:  Function MonthToString( const Month : Integer) : string
5999:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
6000:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
6001:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
6002:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
6003:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
6004:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
6005:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
6006:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
6007:  Function TimeStringToStTime( const Picture, S : string) : TStTime
6008:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6009:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6010:  Function DateStringIsBlank( const Picture, S : string) : Boolean
6011:  Function InternationalDate( ForceCentury : Boolean) : string
6012:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
6013:  Function InternationalTime( ShowSeconds : Boolean) : string
6014:  Procedure ResetInternationalInfo
6015:
6016: procedure SIRegister_StBase(CL: TPSPascalCompiler);
6017:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
6018:  Function AnsiUpperCaseShort32( const S : string) : string
6019:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
6020:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
6021:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
6022:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
6023:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
6024:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
6025:  Function Upcase( C : AnsiChar) : AnsiChar
6026:  Function LoCase( C : AnsiChar) : AnsiChar
6027:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
6028:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
6029:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6030:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6031:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6032:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
6033:  Procedure RaiseContainerError( Code : longint)
6034:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6035:  Function ProductOverflow( A, B : LongInt) : Boolean
6036:  Function StNewStr( S : string) : PShortString
6037:  Procedure StDisposeStr( PS : PShortString)
6038:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
6039:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
6040:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6041:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6042:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6043:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6044:
6045: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6046: begin
6047:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6048:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6049:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6050:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6051:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6052: end;
6053:
6054: //**********unit unit ; StMath Package of SysTools*****************************************
6055: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6056: Function PowerS( Base, Exponent : Extended) : Extended
6057: Function StInvCos( X : Double) : Double
6058: Function StInvSin( Y : Double) : Double
6059: Function StInvTan2( X, Y : Double) : Double
6060: Function StTan( A : Double) : Double
6061: Procedure DumpException;  //unit StExpEng;
6062: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6063:
6064: //**********unit unit ; StCRC Package of SysTools*****************************************
6065: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6066: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6067: Function Adler32OfFile( FileName : AnsiString) : LongInt
6068: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6069: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6070: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6071: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6072: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6073: Function Crc32OfFile( FileName : AnsiString) : LongInt
6074: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6075: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
```

```
6076: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6077: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6078: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6079: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6080:
6081: //**********unit unit ; StBCD Package of SysTools*****************************************
6082: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6083: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6084: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6085: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6086: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6087: Function NegBcd( const B : TbcdS) : TbcdS
6088: Function AbsBcd( const B : TbcdS) : TbcdS
6089: Function FracBcd( const B : TbcdS) : TbcdS
6090: Function IntBcd( const B : TbcdS) : TbcdS
6091: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6092: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6093: Function ValBcd( const S : string) : TbcdS
6094: Function LongBcd( L : LongInt) : TbcdS
6095: Function ExtBcd( E : Extended) : TbcdS
6096: Function ExpBcd( const B : TbcdS) : TbcdS
6097: Function LnBcd( const B : TbcdS) : TbcdS
6098: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6099: Function PowBcd( const B, E : TbcdS) : TbcdS
6100: Function SqrtBcd( const B : TbcdS) : TbcdS
6101: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6102: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6103: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6104: Function IsIntBcd( const B : TbcdS) : Boolean
6105: Function TruncBcd( const B : TbcdS) : LongInt
6106: Function BcdExt( const B : TbcdS) : Extended
6107: Function RoundBcd( const B : TbcdS) : LongInt
6108: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6109: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6110: Function FormatBcd( const Format : string; const B : TbcdS) : string
6111: Function StrGeneralBcd( const B : TbcdS) : string
6112: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6113: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6114:
6115: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6116: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6117: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6118: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6119: Function StDeEscape( const EscStr : AnsiString) : Char
6120: Function StDoEscape( Delim : Char) : AnsiString
6121: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6122: Function AnsiHashText( const S : string; Size : Integer) : Integer
6123: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6124: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6125: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6126:
6127: //**********unit unit ; StNetCon Package of SysTools*****************************************
6128:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6129:     Constructor Create( AOwner : TComponent)
6130:     Function Connect : DWord
6131:     Function Disconnect : DWord
6132:     RegisterProperty('Password', 'String', iptrw);
6133:     Property('UserName', 'String', iptrw);
6134:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6135:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6136:     Property('LocalDevice', 'String', iptrw);
6137:     Property('ServerName', 'String', iptrw);
6138:     Property('ShareName', 'String', iptrw);
6139:     Property('OnConnect', 'TNotifyEvent', iptrw);
6140:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6141:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6142:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6143:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6144:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6145:   end;
6146: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6147: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6148: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6149: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6150: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6151: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6152: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6153: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6154: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6155: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6156: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6157: Function SuspendThread( hThread : THandle) : DWORD
6158: Function ResumeThread( hThread : THandle) : DWORD
6159: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6160: Function GetCurrentThread : THandle
6161: Procedure ExitThread( dwExitCode : DWORD)
6162: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6163: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
```

```
6164: Procedure EndThread(ExitCode: Integer);
6165: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6166: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6167: Procedure FreeProcInstance( Proc : FARPROC)
6168: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6169: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6170: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6171: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6172: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6173: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6174: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6175: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6176: Function CurrentParallelJobInfo : TParallelJobInfo
6177: Function ObtainParallelJobInfo : TParallelJobInfo
6178: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6179: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6180: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6181: Function
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL');
6182: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL');
6183: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD) : BOOL');
6184: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6185: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6186:
6187: ************************************************unit uPSI_JclMime;
6188:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6189:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6190:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6191:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6192:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6193:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6194:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6195:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6196:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6197:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6198:
6199: ************************************************unit uPSI_JclPrint;
6200:  Procedure DirectPrint( const Printer, Data : string)
6201:  Procedure SetPrinterPixelsPerInch
6202:  Function GetPrinterResolution : TPoint
6203:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6204:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6205:
6206:
6207: //************************************unit uPSI_ShLwApi;***********************************
6208:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6209:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6210:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6211:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6212:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6213:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6214:  Function StrDup( lpSrch : PChar) : PChar
6215:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6216:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6217:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6218:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6219: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6220:  Function StrPBrk( psz, pszSet : PChar) : PChar
6221:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6222:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6223:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6224:  Function StrSpn( psz, pszSet : PChar) : Integer
6225:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6226:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6227:  Function StrToInt( lpSrch : PChar) : Integer
6228:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6229:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6230:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6231:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6232:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6233:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6234:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6235:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6236:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6237:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6238:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6239:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6240:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6241:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6242:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6243:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6244:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6245:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6246:  STIF_DEFAULT','LongWord( $00000000);
6247:  STIF_SUPPORT_HEX','LongWord( $00000001);
```

```
6248:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6249:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6250:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6251:  Function PathAddBackslash( pszPath : PChar) : PChar
6252:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6253:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6254:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6255:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6256:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6257:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6258:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6259:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6260:  Function PathFileExists( pszPath : PChar) : BOOL
6261:  Function PathFindExtension( pszPath : PChar) : PChar
6262:  Function PathFindFileName( pszPath : PChar) : PChar
6263:  Function PathFindNextComponent( pszPath : PChar) : PChar
6264:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6265:  Function PathGetArgs( pszPath : PChar) : PChar
6266:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6267:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6268:  Function PathGetCharType( ch : Char) : UINT
6269:  GCT_INVALID','LongWord( $0000);
6270:  GCT_LFNCHAR','LongWord( $0001);
6271:  GCT_SHORTCHAR','LongWord( $0002);
6272:  GCT_WILD','LongWord( $0004);
6273:  GCT_SEPARATOR','LongWord( $0008);
6274:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6275:  Function PathIsDirectory( pszPath : PChar) : BOOL
6276:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6277:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6278:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6279:  Function PathIsRelative( pszPath : PChar) : BOOL
6280:  Function PathIsRoot( pszPath : PChar) : BOOL
6281:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6282:  Function PathIsUNC( pszPath : PChar) : BOOL
6283:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6284:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6285:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6286:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6287:  Function PathIsURL( pszPath : PChar) : BOOL
6288:  Function PathMakePretty( pszPath : PChar) : BOOL
6289:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6290:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6291:  Procedure PathQuoteSpaces( lpsz : PChar)
6292:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6293:  Procedure PathRemoveArgs( pszPath : PChar)
6294:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6295:  Procedure PathRemoveBlanks( pszPath : PChar)
6296:  Procedure PathRemoveExtension( pszPath : PChar)
6297:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6298:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6299:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6300:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6301:  Function PathSkipRoot( pszPath : PChar) : PChar
6302:  Procedure PathStripPath( pszPath : PChar)
6303:  Function PathStripToRoot( pszPath : PChar) : BOOL
6304:  Procedure PathUnquoteSpaces( lpsz : PChar)
6305:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6306:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6307:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6308:  Procedure PathUndecorate( pszPath : PChar)
6309:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6310:  URL_SCHEME_INVALID','LongInt'( - 1);
6311:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6312:  URL_SCHEME_FTP','LongInt'( 1);
6313:  URL_SCHEME_HTTP','LongInt'( 2);
6314:  URL_SCHEME_GOPHER','LongInt'( 3);
6315:  URL_SCHEME_MAILTO','LongInt'( 4);
6316:  URL_SCHEME_NEWS','LongInt'( 5);
6317:  URL_SCHEME_NNTP','LongInt'( 6);
6318:  URL_SCHEME_TELNET','LongInt'( 7);
6319:  URL_SCHEME_WAIS','LongInt'( 8);
6320:  URL_SCHEME_FILE','LongInt'( 9);
6321:  URL_SCHEME_MK','LongInt'( 10);
6322:  URL_SCHEME_HTTPS','LongInt'( 11);
6323:  URL_SCHEME_SHELL','LongInt'( 12);
6324:  URL_SCHEME_SNEWS','LongInt'( 13);
6325:  URL_SCHEME_LOCAL','LongInt'( 14);
6326:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6327:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6328:  URL_SCHEME_ABOUT','LongInt'( 17);
6329:  URL_SCHEME_RES','LongInt'( 18);
6330:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6331:  URL_SCHEME', 'Integer
6332:  URL_PART_NONE','LongInt'( 0);
6333:  URL_PART_SCHEME','LongInt'( 1);
6334:  URL_PART_HOSTNAME','LongInt'( 2);
6335:  URL_PART_USERNAME','LongInt'( 3);
6336:  URL_PART_PASSWORD','LongInt'( 4);
```

```
6337:  URL_PART_PORT','LongInt'( 5);
6338:  URL_PART_QUERY','LongInt'( 6);
6339:  URL_PART', 'DWORD
6340:  URLIS_URL','LongInt'( 0);
6341:  URLIS_OPAQUE','LongInt'( 1);
6342:  URLIS_NOHISTORY','LongInt'( 2);
6343:  URLIS_FILEURL','LongInt'( 3);
6344:  URLIS_APPLIABLE','LongInt'( 4);
6345:  URLIS_DIRECTORY','LongInt'( 5);
6346:  URLIS_HASQUERY','LongInt'( 6);
6347:  TUrlIs', 'DWORD
6348:  URL_UNESCAPE','LongWord( $10000000);
6349:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6350:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6351:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6352:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6353:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6354:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6355:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6356:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6357:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6358:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6359:  URL_INTERNAL_PATH','LongWord( $00800000);
6360:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6361:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6362:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6363:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6364:  URL_APPLY_DEFAULT','LongWord( $00000001);
6365:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6366:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6367:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6368:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6369:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6370:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6371:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6372:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6373:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6374:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6375:  Function UrlGetLocation( psz1 : PChar) : PChar
6376:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6377:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6378:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6379:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6380:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6381:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6382:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6383:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6384:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6385:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6386:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6387:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6388:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6389:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6390:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6391:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6392:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6393:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6394:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6395:  SHREGDEL_DEFAULT','LongWord( $00000000);
6396:  SHREGDEL_HKCU','LongWord( $00000001);
6397:  SHREGDEL_HKLM','LongWord( $00000010);
6398:  SHREGDEL_BOTH','LongWord( $00000011);
6399:  SHREGENUM_DEFAULT','LongWord( $00000000);
6400:  SHREGENUM_HKCU','LongWord( $00000001);
6401:  SHREGENUM_HKLM','LongWord( $00000010);
6402:  SHREGENUM_BOTH','LongWord( $00000011);
6403:  SHREGSET_HKCU','LongWord( $00000001);
6404:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6405:  SHREGSET_HKLM','LongWord( $00000004);
6406:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6407:  TSHRegDelFlags', 'DWORD
6408:  TSHRegEnumFlags', 'DWORD
6409:  HUSKEY', 'THandle
6410:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6411:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6412:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6413:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6414:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6415:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6416:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6417:  ASSOCF_VERIFY','LongWord( $00000040);
6418:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6419:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6420:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6421:  ASSOCF', 'DWORD
6422:  ASSOCSTR_COMMAND','LongInt'( 1);
6423:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6424:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
```

```
6425:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6426:  ASSOCSTR_NOOPEN','LongInt'( 5);
6427:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6428:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6429:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6430:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6431:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6432:  ASSOCSTR_INFOTIP','LongInt'( 11);
6433:  ASSOCSTR_MAX','LongInt'( 12);
6434:  ASSOCSTR', 'DWORD
6435:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6436:  ASSOCKEY_APP','LongInt'( 2);
6437:  ASSOCKEY_CLASS','LongInt'( 3);
6438:  ASSOCKEY_BASECLASS','LongInt'( 4);
6439:  ASSOCKEY_MAX','LongInt'( 5);
6440:  ASSOCKEY', 'DWORD
6441:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6442:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6443:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6444:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6445:  ASSOCDATA_MAX','LongInt'( 5);
6446:  ASSOCDATA', 'DWORD
6447:  ASSOCENUM_NONE','LongInt'( 0);
6448:  ASSOCENUM', 'DWORD
6449:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6450:  SHACF_DEFAULT $00000000);
6451:  SHACF_FILESYSTEM','LongWord( $00000001);
6452:  SHACF_URLHISTORY','LongWord( $00000002);
6453:  SHACF_URLMRU','LongWord( $00000004);
6454:  SHACF_USETAB','LongWord( $00000008);
6455:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6456:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6457:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6458:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6459:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6460:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6461:  Procedure SHSetThreadRef( punk : IUnknown)
6462:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6463:  CTF_INSIST','LongWord( $00000001);
6464:  CTF_THREAD_REF','LongWord( $00000002);
6465:  CTF_PROCESS_REF','LongWord( $00000004);
6466:  CTF_COINIT','LongWord( $00000008);
6467:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6468:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6469:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6470:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6471:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6472:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6473:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6474:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6475:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6476:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6477:  Function SetRectEmpty( var lprc : TRect) : BOOL
6478:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6479:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6480:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6481:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6482:
6483:  Function InitializeFlatSB( hWnd : HWND) : Bool
6484:  Procedure UninitializeFlatSB( hWnd : HWND)
6485:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6486:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6487:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6488:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6489:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6490:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6491:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6492:
6493:
6494: //  ****************************************** 204 unit uPSI_ShellAPI;
6495: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6496: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6497: Procedure DragFinish( Drop : HDROP)
6498: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6499: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6500: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6501: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6502: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6503: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6504: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6505: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6506: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6507: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6508: Procedure SHFreeNameMappings( hNameMappings : THandle)
6509:
6510:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6511:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6512:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6513:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
```

```
6514:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6515:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6516:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6517:  Function SimpleXMLEncode( const S : string) : string
6518:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6519:  Function XMLEncode( const S : string) : string
6520:  Function XMLDecode( const S : string) : string
6521:  Function EntityEncode( const S : string) : string
6522:  Function EntityDecode( const S : string) : string
6523:
6524: procedure RIRegister_CPort_Routines(S: TPSExec);
6525:  Procedure EnumComPorts( Ports : TStrings)
6526:  Procedure ListComPorts( Ports : TStrings)
6527:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6528:  Function GetComPorts: TStringlist;
6529:  Function StrToBaudRate( Str : string) : TBaudRate
6530:  Function StrToStopBits( Str : string) : TStopBits
6531:  Function StrToDataBits( Str : string) : TDataBits
6532:  Function StrToParity( Str : string) : TParityBits
6533:  Function StrToFlowControl( Str : string) : TFlowControl
6534:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6535:  Function StopBitsToStr( StopBits : TStopBits) : string
6536:  Function DataBitsToStr( DataBits : TDataBits) : string
6537:  Function ParityToStr( Parity : TParityBits) : string
6538:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6539:  Function ComErrorsToStr( Errors : TComErrors) : String
6540:
6541:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6542:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6543:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6544:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6545:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6546:  Function GetMessagePos : DWORD
6547:  Function GetMessageTime : Longint
6548:  Function GetMessageExtraInfo : Longint
6549:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6550:  Procedure JAddToRecentDocs( const Filename : string)
6551:  Procedure ClearRecentDocs
6552:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6553:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6554:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6555:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6556:  Function RecycleFile( FileToRecycle : string) : Boolean
6557:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6558:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6559:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6560:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6561:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6562:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6563:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6564:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6565:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6566:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6567:
6568: ****************************************** unit uPSI_JclPeImage;
6569:
6570:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6571: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6572:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6573:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6574:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6575:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6576:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6577:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6578:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6579:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Bool
6580:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6581:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6582:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6583:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
      LibraryName:string; IncludeLibNames : Boolean): Boolean
6584:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6585:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
```

```
6586:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6587:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
       NamesList:TStrings):Bool
6588:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6589:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
       Descript:Bool):Bool;
6590:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6591:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6592:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6593:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6594:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6595:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6596:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
       PImageSectionHeader
6597:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6598:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6599:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
       ___Pointer;
6600:   SIRegister_TJclPeSectionStream(CL);
6601:   SIRegister_TJclPeMapImgHookItem(CL);
6602:   SIRegister_TJclPeMapImgHooks(CL);
6603:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6604:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6605:   Type TJclBorUmSymbolKind', '(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6606:   TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6607:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6608:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6609:   TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6610:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6611:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6612:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6613:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6614:  Function PeBorUnmangleName3( const Name : string) : string;
6615:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6616:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6617:
6618:
6619: //***************** SysTools uPSI_StSystem; ************************************
6620:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6621:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6622:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6623:  //Procedure EnumerateDirectories(const
       StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6624:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6625:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6626:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6627:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6628:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6629:  Function FlushOsBuffers( Handle : Integer) : Boolean
6630:  Function GetCurrentUser : AnsiString
6631:  Function GetDiskClass( Drive : Char) : DiskClass
6632:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6633:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6634:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6635:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6636:  Function getDiskSpace2(const path: String; index: integer): int64;
6637:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6638:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6639:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6640:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6641:  Function GetLongPath( const APath : AnsiString) : AnsiString
6642:  Function GetMachineName : AnsiString
6643:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6644:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6645:  Function GetShortPath( const APath : AnsiString) : AnsiString
6646:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6647:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6648:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6649:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6650:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6651:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6652:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6653:  Function IsDriveReady( Drive : Char) : Boolean
6654:  Function IsFile( const FileName : AnsiString) : Boolean
6655:  Function IsFileArchive( const S : AnsiString) : Integer
6656:  Function IsFileHidden( const S : AnsiString) : Integer
6657:  Function IsFileReadOnly( const S : AnsiString) : Integer
6658:  Function IsFileSystem( const S : AnsiString) : Integer
6659:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6660:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6661:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6662:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
```

```
6663:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6664:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6665:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6666:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6667:  Function ValidDrive( Drive : Char) : Boolean
6668:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6669:
6670: //****************************unit uPSI_JclLANMan;*****************************************
6671: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6672:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6673:  Function DeleteAccount( const Servername, Username : string) : Boolean
6674:  Function DeleteLocalAccount( Username : string) : Boolean
6675:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6676:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6677:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6678:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6679:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6680:  Function LocalGroupExists( const Group : string) : Boolean
6681:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6682:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6683:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6684:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6685:  Function IsLocalAccount( const AccountName : string) : Boolean
6686:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6687:  Function GetRandomString( NumChar : cardinal) : string
6688:
6689: //****************************unit uPSI_cUtils;*****************************************
6690: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6691: Function cIsWinNT : boolean
6692:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6693:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6694:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6695:  Function cGetShortName( FileName : string) : string
6696:  Procedure cShowError( Msg : String)
6697:  Function cCommaStrToStr( s : string; formatstr : string) : string
6698:  Function cIncludeQuoteIfSpaces( s : string) : string
6699:  Function cIncludeQuoteIfNeeded( s : string) : string
6700:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6701:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6702:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6703:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6704:  Function cCodeInstoStr( s : string) : string
6705:  Function cStrtoCodeIns( s : string) : string
6706:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6707:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6708:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6709:  Function cPointtoStr( const pt : TPoint) : string
6710:  Function cListtoStr( const List : TStrings) : string
6711:  Function ListtoStr( const List : TStrings) : string
6712:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6713:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6714:  Function cGetFileTyp( const FileName : string) : TUnitType
6715:  Function cGetExTyp( const FileName : string) : TExUnitType
6716:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6717:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6718:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6719:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6720:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6721:  Function cGenMakePath( FileName : String) : String;
6722:  Function cGenMakePath2( FileName : String) : String
6723:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6724:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6725:  Function cCalcMod( Count : Integer) : Integer
6726:  Function cGetVersionString( FileName : string) : string
6727:  Function cCheckChangeDir( var Dir : string) : boolean
6728:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6729:  Function cIsNumeric( s : string) : boolean
6730:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6731:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6732:  Function GetFileTyp( const FileName : string) : TUnitType
6733:  Function Atoi(const aStr: string): integer
6734:  Function Itoa(const aint: integer): string
6735:  Function Atof(const aStr: string): double');
6736:  Function Atol(const aStr: string): longint');
6737:
6738:
6739: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6740: begin
6741:   FindClass('TOBJECT'),'EHTTP
6742:   FindClass('TOBJECT'),'EHTTPParser
6743:   //AnsiCharSet', 'set of AnsiChar
6744:   AnsiStringArray', 'array of AnsiString
6745:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6746:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6747:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
```

```
6748:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6749:    +'CustomMinVersion : Integer; end
6750:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6751:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6752:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6753:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6754:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6755:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6756:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6757:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6758:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6759:    +'nection, hntOrigin, hntKeepAlive )
6760:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6761:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6762:    +' AnsiString; end
6763:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6764:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6765:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6766:   //PHTTPContentLength', '^THTTPContentLength // will not work
6767:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6768:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6769:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6770:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6771:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6772:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6773:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6774:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6775:    +' CustomStr : AnsiString; end
6776:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6777:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6778:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6779:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6780:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6781:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6782:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6783:    +'m; Custom : AnsiString; end
6784:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6785:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6786:    +' Custom : AnsiString; end
6787:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6788:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6789:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6790:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6791:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6792:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6793:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6794:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6795:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6796:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6797:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6798:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6799:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6800:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6801:    +'FieldEnum; List : array of THTTPContentEncoding; end
6802:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6803:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6804:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6805:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6806:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6807:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6808:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6809:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6810:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6811:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6812:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6813:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6814:    +'CustomFieldArray; Custom : AnsiString; end
6815:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6816:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6817:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6818:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6819:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6820:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6821:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6822:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6823:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6824:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6825:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6826:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6827:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6828:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6829:   THTTPFixedHeaders','array[0..42] of AnsiString
6830:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6831:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6832:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6833:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6834:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6835:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6836:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
```

```
6837:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6838:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6839:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6840:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6841:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6842:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6843:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6844:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6845:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6846:    +' THTTPDateField; Age : THTTPAgeField; end
6847:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6848:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6849:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6850:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6851:  Procedure InitHTTPRequest( var A : THTTPRequest)
6852:  Procedure InitHTTPResponse( var A : THTTPResponse)
6853:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6854:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6855:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6856:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6857:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6858:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6859:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6860:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6861:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6862:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6863:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6864:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6865:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6866:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6867:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6868:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6869:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6870:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6871:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6872:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6873:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6874:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6875:   THTTPStringOption', '( hsoNone )
6876:   THTTPStringOptions', 'set of THTTPStringOption
6877:   FindClass('TOBJECT'),'TAnsiStringBuilder
6878:
6879:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6880:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6881:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6882:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6883:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6884:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6885:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6886:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6887:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6888:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6889:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6890:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6891:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6892:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6893:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6894:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6895:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6896:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6897:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6898:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6899:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6900:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6901:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6902:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
```

```
6903:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6904:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6905:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6906:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
       TAnsiStringBuilder; const P : THTTPStringOptions)
6907:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
       THTTPStrOptions);
6908:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
6909:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
       P:THTTPStringOptions);
6910:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6911:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6912:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6913:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6914:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6915:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6916:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
       Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6917:     +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6918:   SIRegister_THTTPParser(CL);
6919:   FindClass('TOBJECT'),'THTTPContentDecoder
6920:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6921:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6922:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6923:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6924:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6925:   SIRegister_THTTPContentDecoder(CL);
6926:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6927:   FindClass('TOBJECT'),'THTTPContentReader
6928:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6929:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
       LogLevel:Int;
6930:   SIRegister_THTTPContentReader(CL);
6931:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6932:   FindClass('TOBJECT'),'THTTPContentWriter
6933:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6934:   SIRegister_THTTPContentWriter(CL);
6935:  Procedure SelfTestcHTTPUtils
6936: end;
6937:
6938: (*----------------------------------------------------------------------------*)
6939: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6940: begin
6941:  'TLSLibraryVersion','String '1.00
6942:  'TLSError_None','LongInt'( 0);
6943:  'TLSError_InvalidBuffer','LongInt'( 1);
6944:  'TLSError_InvalidParameter','LongInt'( 2);
6945:  'TLSError_InvalidCertificate','LongInt'( 3);
6946:  'TLSError_InvalidState','LongInt'( 4);
6947:  'TLSError_DecodeError','LongInt'( 5);
6948:  'TLSError_BadProtocol','LongInt'( 6);
6949:  Function TLSErrorMessage( const TLSError : Integer) : String
6950:   SIRegister_ETLSError(CL);
6951:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6952:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6953:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6954:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6955:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6956:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6957:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6958:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6959:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6960:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6961:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6962:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6963:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6964:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6965:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6966:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6967:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6968:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6969:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6970:   PTLSRandom', '^TTLSRandom // will not work
6971:  Procedure InitTLSRandom( var Random : TTLSRandom)
6972:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6973:  'TLSSessionIDMaxLen','LongInt'( 32);
6974:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6975:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6976:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6977:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6978:    +'; Signature : TTLSSignatureAlgorithm; end
6979:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6980:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6981:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6982:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
```

```
6983:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6984:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6985:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6986:    +'nteger; Supported : Boolean; end
6987:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6988:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6989:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6990:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6991:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6992:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6993:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6994:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6995:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6996:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6997:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6998:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6999:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
7000:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
7001:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
7002:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
7003:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
7004:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
7005:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
7006:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
7007:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
7008:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
7009:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
7010:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
7011:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
7012:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7013:  Procedure SelfTestcTLSUtils
7014: end;
7015:
7016: (*----------------------------------------------------------------------------*)
7017: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
7018: begin
7019:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
7020:  // pBoard', '^tBoard // will not work
7021:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
7022:  Function rCheckMove( color : byte; cx, cy : integer) : integer
7023:  //Function rDoStep( data : pBoard) : word
7024:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
7025: end;
7026:
7027: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7028: begin
7029: Function InEditMode( ADataset : TDataset) : Boolean
7030: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
7031: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7032: Function GetFieldText( AField : TField) : String
7033: end;
7034:
7035: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
7036: begin
7037:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7038:   TMyPrintRange', '( prAll, prSelected )
7039:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7040:    +'ded, ssDateTime, ssTime, ssCustom )
7041:   TSortDirection', '( sdAscending, sdDescending )
7042:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7043:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7044:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7045:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7046:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7047:   SIRegister_TSortOptions(CL);
7048:   SIRegister_TPrintOptions(CL);
7049:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7050:   SIRegister_TSortedList(CL);
7051:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7052:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7053:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7054:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7055:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7056:   SIRegister_TFontSetting(CL);
7057:   SIRegister_TFontList(CL);
7058:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7059:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7060:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7061:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7062:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
```

```
7063:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7064:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7065:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7066:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
7067:    +'r; var SortStyle : TSortStyle)
7068:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
7069:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7070:   SIRegister_TSortGrid(CL);
7071:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7072:  Function NormalCompare( const Str1, Str2 : String) : Integer
7073:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7074:  Function NumericCompare( const Str1, Str2 : String) : Integer
7075:  Function TimeCompare( const Str1, Str2 : String) : Integer
7076:  //Function Compare( Item1, Item2 : Pointer) : Integer
7077: end;
7078:
7079: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7080:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7081:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7082:  Procedure IBDataBaseError
7083:  Function StatusVector : PISC_STATUS
7084:  Function StatusVectorArray : PStatusVector
7085:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7086:  Function StatusVectorAsText : string
7087:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7088:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7089:
7090:
7091: //****************************unit uPSI_BoldUtils;****************************************
7092:  Function CharCount( c : char; const s : string) : integer
7093:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7094:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7095:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7096:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7097:  Function BoldTrim( const S : string) : string
7098:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7099:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7100:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7101:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7102:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7103:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7104:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7105:  Function CapitalisedToSpaced( Capitalised : String) : String
7106:  Function SpacedToCapitalised( Spaced : String) : String
7107:  Function BooleanToString( BoolValue : Boolean) : String
7108:  Function StringToBoolean( StrValue : String) : Boolean
7109:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7110:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7111:  Function StringListToVarArray( List : TStringList) : variant
7112:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7113:  Function GetComputerNameStr : string
7114:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7115:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7116:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7117:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7118:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7119:  Procedure EnsureTrailing( var Str : String; ch : char)
7120:  Function BoldDirectoryExists( const Name : string) : Boolean
7121:  Function BoldForceDirectories( Dir : string) : Boolean
7122:  Function BoldRootRegistryKey : string
7123:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7124:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7125:  Function LogicalAnd( A, B : Integer) : Boolean
7126:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7127:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7128:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7129:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7130:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7131:  Function IsFirstInstance : Boolean
7132:  Procedure ActivateFirst( AString : PChar)
7133:  Procedure ActivateFirstCommandLine
7134:  function MakeAckPkt(const BlockNumber: Word): string;
7135:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7136:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7137:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7138:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7139:  function IdStrToWord(const Value: String): Word;
7140:  function IdWordToStr(const Value: Word): WordStr;
7141:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7142:  Function CPUFeatures : TCPUFeatures
7143:
7144: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7145: begin
7146:   AddTypeS('TXRTLBitIndex', 'Integer
7147:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7148:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7149:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7150:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
```

```
7151:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7152:  Function XRTLSwapHiLo16( X : Word) : Word
7153:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7154:  Function XRTLSwapHiLo64( X : Int64) : Int64
7155:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7156:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7157:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7158:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7159:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7160:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7161:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7162:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7163:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7164:  Function XRTLPopulation( A : Cardinal) : Cardinal
7165: end;
7166:
7167: Function XRTLURLDecode( const ASrc : WideString) : WideString
7168: Function XRTLURLEncode( const ASrc : WideString) : WideString
7169: Function XRTLURINormalize( const AURI : WideString) : WideString
7170: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7171: Function XRTLExtractLongPathName(APath: string): string;
7172:
7173: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7174: begin
7175:   AddTypeS('Int8', 'ShortInt
7176:   AddTypeS('Int16', 'SmallInt
7177:   AddTypeS('Int32', 'LongInt
7178:   AddTypeS('UInt8', 'Byte
7179:   AddTypeS('UInt16', 'Word
7180:   AddTypeS('UInt32', 'LongWord
7181:   AddTypeS('UInt64', 'Int64
7182:   AddTypeS('Word8', 'UInt8
7183:   AddTypeS('Word16', 'UInt16
7184:   AddTypeS('Word32', 'UInt32
7185:   AddTypeS('Word64', 'UInt64
7186:   AddTypeS('LargeInt', 'Int64
7187:   AddTypeS('NativeInt', 'Integer
7188:   AddTypeS('NativeUInt', 'Cardinal
7189:   Const('BitsPerByte','LongInt'( 8);
7190:   Const('BitsPerWord','LongInt'( 16);
7191:   Const('BitsPerLongWord','LongInt'( 32);
7192:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7193:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7194:  Function MinI( const A, B : Integer) : Integer
7195:  Function MaxI( const A, B : Integer) : Integer
7196:  Function MinC( const A, B : Cardinal) : Cardinal
7197:  Function MaxC( const A, B : Cardinal) : Cardinal
7198:  Function SumClipI( const A, I : Integer) : Integer
7199:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7200:  Function InByteRange( const A : Int64) : Boolean
7201:  Function InWordRange( const A : Int64) : Boolean
7202:  Function InLongWordRange( const A : Int64) : Boolean
7203:  Function InShortIntRange( const A : Int64) : Boolean
7204:  Function InSmallIntRange( const A : Int64) : Boolean
7205:  Function InLongIntRange( const A : Int64) : Boolean
7206:   AddTypeS('Bool8', 'ByteBool
7207:   AddTypeS('Bool16', 'WordBool
7208:   AddTypeS('Bool32', 'LongBool
7209:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7210:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7211:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7212:  Const('MinSingle','Single').setExtended( 1.5E-45);
7213:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7214:  Const('MinDouble','Double').setExtended( 5.0E-324);
7215:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7216:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7217:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7218:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7219:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7220:  Function MinF( const A, B : Float) : Float
7221:  Function MaxF( const A, B : Float) : Float
7222:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7223:  Function InSingleRange( const A : Float) : Boolean
7224:  Function InDoubleRange( const A : Float) : Boolean
7225:  Function InCurrencyRange( const A : Float) : Boolean;
7226:  Function InCurrencyRange1( const A : Int64) : Boolean;
7227:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7228:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7229:  Function FloatIsInfinity( const A : Extended) : Boolean
7230:  Function FloatIsNaN( const A : Extended) : Boolean
7231:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7232:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7233:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7234:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7235:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7236:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7237:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7238:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
```

```
7239:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7240:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7241:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7242:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7243:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7244:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7245:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7246:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7247:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7248:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7249:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7250:  Function SetBitScanForward( const Value : LongWord) : Integer;
7251:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7252:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7253:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7254:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7255:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7256:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7257:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7258:  Function cReverseBits( const Value : LongWord) : LongWord;
7259:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7260:  Function cSwapEndian( const Value : LongWord) : LongWord
7261:  Function cTwosComplement( const Value : LongWord) : LongWord
7262:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7263:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7264:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7265:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7266:  Function cBitCount( const Value : LongWord) : LongWord
7267:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7268:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7269:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7270:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7271:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7272:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7273:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7274:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7275:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7276: //  AddTypeS('CharSet', 'set of AnsiChar
7277:   AddTypeS('CharSet', 'set of Char       //!!!
7278:   AddTypeS('AnsiCharSet', 'TCharSet
7279:   AddTypeS('ByteSet', 'set of Byte
7280:   AddTypeS('AnsiChar', 'Char
7281:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7282:  Function AsByteSet( const C : array of Byte) : ByteSet
7283:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7284:  Procedure ClearCharSet( var C : CharSet)
7285:  Procedure FillCharSet( var C : CharSet)
7286:  procedure FillCharSearchRec;  // with 0
7287:  Procedure ComplementCharSet( var C : CharSet)
7288:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7289:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7290:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7291:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7292:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7293:  Function IsSubSet( const A, B : CharSet) : Boolean
7294:  Function IsEqual( const A, B : CharSet) : Boolean
7295:  Function IsEmpty( const C : CharSet) : Boolean
7296:  Function IsComplete( const C : CharSet) : Boolean
7297:  Function cCharCount( const C : CharSet) : Integer
7298:  Procedure ConvertCaseInsensitive( var C : CharSet)
7299:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7300:  Function IntRangeLength( const Low, High : Integer) : Int64
7301:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7302:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7303:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7304:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7305:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7306:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7307:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7308:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7309:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7310:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7311:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7312:   AddTypeS('UnicodeChar', 'WideChar
7313:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7314:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7315:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7316:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7317:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7318:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7319:  Function cSgn( const A : LongInt) : Integer;
7320:  Function cSgn1( const A : Int64) : Integer;
7321:  Function cSgn2( const A : Extended) : Integer;
7322:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7323:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7324:  Function WideCharToInt( const A : WideChar) : Integer
7325:  Function CharToInt( const A : Char) : Integer
7326:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7327:  Function IntToWideChar( const A : Integer) : WideChar
```

```
7328:  Function IntToChar( const A : Integer) : Char
7329:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7330:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7331:  Function IsHexChar( const Ch : Char) : Boolean
7332:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7333:  Function HexWideCharToInt( const A : WideChar) : Integer
7334:  Function HexCharToInt( const A : Char) : Integer
7335:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7336:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7337:  Function IntToUpperHexChar( const A : Integer) : Char
7338:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7339:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7340:  Function IntToLowerHexChar( const A : Integer) : Char
7341:  Function IntToStringA( const A : Int64) : AnsiString
7342:  Function IntToStringW( const A : Int64) : WideString
7343:  Function IntToString( const A : Int64) : String
7344:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7345:  Function UIntToStringW( const A : NativeUInt) : WideString
7346:  Function UIntToString( const A : NativeUInt) : String
7347:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7348:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7349:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7350:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7351:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7352:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7353:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7354:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7355:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7356:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7357:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7358:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7359:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7360:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7361:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7362:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7363:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7364:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7365:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7366:  Function StringToInt64A( const S : AnsiString) : Int64
7367:  Function StringToInt64W( const S : WideString) : Int64
7368:  Function StringToInt64( const S : String) : Int64
7369:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7370:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7371:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7372:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7373:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7374:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7375:  Function StringToIntA( const S : AnsiString) : Integer
7376:  Function StringToIntW( const S : WideString) : Integer
7377:  Function StringToInt( const S : String) : Integer
7378:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7379:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7380:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7381:  Function StringToLongWordA( const S : AnsiString) : LongWord
7382:  Function StringToLongWordW( const S : WideString) : LongWord
7383:  Function StringToLongWord( const S : String) : LongWord
7384:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7385:  Function HexToUIntW( const S : WideString) : NativeUInt
7386:  Function HexToUInt( const S : String) : NativeUInt
7387:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7388:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7389:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7390:  Function HexToLongWordA( const S : AnsiString) : LongWord
7391:  Function HexToLongWordW( const S : WideString) : LongWord
7392:  Function HexToLongWord( const S : String) : LongWord
7393:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7394:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7395:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7396:  Function OctToLongWordA( const S : AnsiString) : LongWord
7397:  Function OctToLongWordW( const S : WideString) : LongWord
7398:  Function OctToLongWord( const S : String) : LongWord
7399:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7400:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7401:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7402:  Function BinToLongWordA( const S : AnsiString) : LongWord
7403:  Function BinToLongWordW( const S : WideString) : LongWord
7404:  Function BinToLongWord( const S : String) : LongWord
7405:  Function FloatToStringA( const A : Extended) : AnsiString
7406:  Function FloatToStringW( const A : Extended) : WideString
7407:  Function FloatToString( const A : Extended) : String
7408:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7409:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7410:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7411:  Function StringToFloatA( const A : AnsiString) : Extended
7412:  Function StringToFloatW( const A : WideString) : Extended
7413:  Function StringToFloat( const A : String) : Extended
7414:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7415:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7416:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
```

```
7417:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7418:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7419:  unit uPSI_cFundamentUtils;
7420:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7421:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7422:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7423:  Const('CCHARSET','Stringb64_XXEncode);
7424:  Const('CHEXSET','String'0123456789ABCDEF
7425:  Const('HEXDIGITS','String'0123456789ABCDEF
7426:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7427:  Const('DIGISET','String'0123456789
7428:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7429:  Const('DIGISET2','TCharset').SetSet('0123456789'
7430:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7431:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7432:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7433:  Const('NUMBERS','String'0123456789');
7434:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7435:  Function CharSetToStr( const C : CharSet) : AnsiString
7436:  Function StrToCharSet( const S : AnsiString) : CharSet
7437:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7438:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7439:  Function UUDecode( const S : AnsiString) : AnsiString
7440:  Function XXDecode( const S : AnsiString) : AnsiString
7441:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7442:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7443:  Function InterfaceToStrW( const I : IInterface) : WideString
7444:  Function InterfaceToStr( const I : IInterface) : String
7445:  Function ObjectClassName( const O : TObject) : String
7446:  Function ClassClassName( const C : TClass) : String
7447:  Function ObjectToStr( const O : TObject) : String
7448:  Function ObjectToString( const O : TObject) : String
7449:  Function CharSetToStr( const C : CharSet) : AnsiString
7450:  Function StrToCharSet( const S : AnsiString) : CharSet
7451:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7452:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7453:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7454:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7455:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7456:  Const('Bytes1KB','LongInt'( 1024);
7457:   SIRegister_IInterface(CL);
7458:  Procedure SelfTestCFundamentUtils
7459:
7460: Function CreateSchedule : IJclSchedule
7461: Function NullStamp : TTimeStamp
7462: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7463: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7464: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7465:
7466: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7467: begin
7468:  AddTypeS('TFunc', 'function(X : Float) : Float;
7469: Function InitGraphics( Width, Height : Integer) : Boolean
7470: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7471: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7472: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7473: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7474: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7475: Procedure SetGraphTitle( Title : String)
7476: Procedure SetOxTitle( Title : String)
7477: Procedure SetOyTitle( Title : String)
7478: Function GetGraphTitle : String
7479: Function GetOxTitle : String
7480: Function GetOyTitle : String
7481: Procedure PlotOxAxis( Canvas : TCanvas)
7482: Procedure PlotOyAxis( Canvas : TCanvas)
7483: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7484: Procedure WriteGraphTitle( Canvas : TCanvas)
7485: Function SetMaxCurv( NCurv : Byte) : Boolean
7486: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7487: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7488: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7489: Procedure SetCurvStep( CurvIndex, Step : Integer)
7490: Function GetMaxCurv : Byte
7491: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7492: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7493: Function GetCurvLegend( CurvIndex : Integer) : String
7494: Function GetCurvStep( CurvIndex : Integer) : Integer
7495: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7496: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7497: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7498: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7499: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7500: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7501: Function Xpixel( X : Float) : Integer
```

```
7502: Function Ypixel( Y : Float) : Integer
7503: Function Xuser( X : Integer) : Float
7504: Function Yuser( Y : Integer) : Float
7505: end;
7506:
7507: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7508: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7509: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7510: Procedure FFT_Integer_Cleanup
7511: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7512: //unit uPSI_JclStreams;
7513: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7514: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7515: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7516: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7517:
7518: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7519: begin
7520:  FindClass('TOBJECT'),'EInvalidDest
7521:  FindClass('TOBJECT'),'EFCantMove
7522:  Procedure fmxCopyFile( const FileName, DestName : string)
7523:  Procedure fmxMoveFile( const FileName, DestName : string)
7524:  Function fmxGetFileSize( const FileName : string) : LongInt
7525:  Function fmxFileDateTime( const FileName : string) : TDateTime
7526:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7527:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7528: end;
7529:
7530: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7531: begin
7532:  SIRegister_IFindFileIterator(CL);
7533:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7534: end;
7535:
7536: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7537: begin
7538:  Function SkipWhite( cp : PChar) : PChar
7539:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7540:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7541:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7542: end;
7543:
7544: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7545: begin
7546:   SIRegister_TStringHashMapTraits(CL);
7547:  Function CaseSensitiveTraits : TStringHashMapTraits
7548:  Function CaseInsensitiveTraits : TStringHashMapTraits
7549:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7550:    +'e; Right : PHashNode; end
7551:   //PHashArray', '^THashArray // will not work
7552:   SIRegister_TStringHashMap(CL);
7553:  THashValue', 'Cardinal
7554:  Function StrHash( const s : string) : THashValue
7555:  Function TextHash( const s : string) : THashValue
7556:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7557:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7558:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7559:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7560:   SIRegister_TCaseSensitiveTraits(CL);
7561:   SIRegister_TCaseInsensitiveTraits(CL);
7562:
7563:
7564: //************************************************************************unit uPSI_umath;
7565:  Function uExpo( X : Float) : Float
7566:  Function uExp2( X : Float) : Float
7567:  Function uExp10( X : Float) : Float
7568:  Function uLog( X : Float) : Float
7569:  Function uLog2( X : Float) : Float
7570:  Function uLog10( X : Float) : Float
7571:  Function uLogA( X, A : Float) : Float
7572:  Function uIntPower( X : Float; N : Integer): Float
7573:  Function uPower( X, Y : Float) : Float
7574:  Function SgnGamma( X : Float) : Integer
7575:  Function Stirling( X : Float) : Float
7576:  Function StirLog( X : Float) : Float
7577:  Function Gamma( X : Float) : Float
7578:  Function LnGamma( X : Float) : Float
7579:  Function DiGamma( X : Float) : Float
7580:  Function TriGamma( X : Float) : Float
7581:  Function IGamma( X : Float) : Float
7582:  Function JGamma( X : Float) : Float
7583:  Function InvGamma( X : Float) : Float
7584:  Function Erf( X : Float) : Float
7585:  Function Erfc( X : Float) : Float
7586: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7587: { Correlation coefficient between samples X and Y }
7588: function DBeta(A, B, X : Float) : Float;
7589: { Density of Beta distribution with parameters A and B }
7590: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
```

```
7591: Function Beta(X, Y : Float) : Float
7592: Function Binomial( N, K : Integer) : Float
7593: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7594: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7595: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7596: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7597: Function DNorm( X : Float) : Float
7598:
7599: function DGamma(A, B, X : Float) : Float;
7600: { Density of Gamma distribution with parameters A and B }
7601: function DKhi2(Nu : Integer; X : Float) : Float;
7602: { Density of Khi-2 distribution with Nu d.o.f. }
7603: function DStudent(Nu : Integer; X : Float) : Float;
7604: { Density of Student distribution with Nu d.o.f. }
7605: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7606: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7607: function IBeta(A, B, X : Float) : Float;
7608: { Incomplete Beta function}
7609: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7610:
7611: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7612: begin
7613:  Procedure SetOptAlgo( Algo : TOptAlgo)
7614: procedure SetOptAlgo(Algo : TOptAlgo);
7615: { ---------------------------------------------------------------------
7616:   Sets the optimization algorithm according to Algo, which must be
7617:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7618:
7619:  Function GetOptAlgo : TOptAlgo
7620:  Procedure SetMaxParam( N : Byte)
7621:  Function GetMaxParam : Byte
7622:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7623:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7624:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7625:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
     Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7626:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
     MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7627:  Procedure SetMCFile( FileName : String)
7628:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7629:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
     LastPar:Integer;V:TMatrix);
7630: end;
7631:
7632: (*----------------------------------------------------------------------------*)
7633: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7634: begin
7635:  Procedure SaveSimplex( FileName : string)
7636:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7637: end;
7638: (*----------------------------------------------------------------------------*)
7639: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7640: begin
7641:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7642:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7643: end;
7644:
7645: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7646: begin
7647:  Function LTrim( S : String) : String
7648:  Function RTrim( S : String) : String
7649:  Function uTrim( S : String) : String
7650:  Function StrChar( N : Byte; C : Char) : String
7651:  Function RFill( S : String; L : Byte) : String
7652:  Function LFill( S : String; L : Byte) : String
7653:  Function CFill( S : String; L : Byte) : String
7654:  Function Replace( S : String; C1, C2 : Char) : String
7655:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7656:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7657:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7658:  Function FloatStr( X : Float) : String
7659:  Function IntStr( N : LongInt) : String
7660:  Function uCompStr( Z : Complex) : String
7661: end;
7662:
7663: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7664: begin
7665:  Function uSinh( X : Float) : Float
7666:  Function uCosh( X : Float) : Float
7667:  Function uTanh( X : Float) : Float
7668:  Function uArcSinh( X : Float) : Float
7669:  Function uArcCosh( X : Float) : Float
7670:  Function ArcTanh( X : Float) : Float
7671:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7672: end;
7673:
7674: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7675: begin
7676: type RNG_Type =
```

```
7677:   (RNG_MWC,      { Multiply-With-Carry }
7678:    RNG_MT,       { Mersenne Twister }
7679:    RNG_UVAG);    { Universal Virtual Array Generator }
7680:  Procedure SetRNG( RNG : RNG_Type)
7681:  Procedure InitGen( Seed : RNG_IntType)
7682:  Procedure SRand( Seed : RNG_IntType)
7683:  Function IRanGen : RNG_IntType
7684:  Function IRanGen31 : RNG_IntType
7685:  Function RanGen1 : Float
7686:  Function RanGen2 : Float
7687:  Function RanGen3 : Float
7688:  Function RanGen53 : Float
7689: end;
7690:
7691: //  Optimization by Simulated Annealing
7692: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7693: begin
7694:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7695:  Procedure SA_CreateLogFile( FileName : String)
7696:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7697: end;
7698:
7699: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7700: begin
7701:  Procedure InitUVAGbyString( KeyPhrase : string)
7702:  Procedure InitUVAG( Seed : RNG_IntType)
7703:  Function IRanUVAG : RNG_IntType
7704: end;
7705:
7706: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7707: begin
7708:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7709:  Procedure GA_CreateLogFile( LogFileName : String)
7710:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7711: end;
7712:
7713:  TVector', 'array of Float
7714: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7715: begin
7716:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7717:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7718: end;
7719:
7720: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7721: begin
7722:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7723:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7724: end;
7725:
7726: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7727: begin
7728:   FT_Result', 'Integer
7729:   //TDWordptr', '^DWord // will not work
7730:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7731:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7732:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7733:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7734:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7735:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7736:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7737:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7738:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7739:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7740:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7741:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7742:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7743:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7744:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7745:    yte; end
7746: end;
7747:
7748:
7749: //************************************ PaintFX*************************
7750: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7751: begin
7752:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7753:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7754:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7755:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7756:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7757:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7758:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7759:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7760:     Procedure Turn( Src, Dst : TBitmap)
7761:     Procedure TurnRight( Src, Dst : TBitmap)
7762:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7763:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7764:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7765:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
```

```
7766:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7767:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7768:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7769:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7770:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7771:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7772:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7773:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7774:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7775:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7776:     Procedure Emboss( var Bmp : TBitmap)
7777:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7778:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7779:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7780:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7781:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7782:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7783:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7784:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7785:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7786:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7787:     Procedure SemiOpaque( Src, Dst : TBitmap)
7788:     Procedure ShadowDownLeft( const Dst : TBitmap)
7789:     Procedure ShadowDownRight( const Dst : TBitmap)
7790:     Procedure ShadowUpLeft( const Dst : TBitmap)
7791:     Procedure ShadowUpRight( const Dst : TBitmap)
7792:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7793:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7794:     Procedure FlipRight( const Dst : TBitmap)
7795:     Procedure FlipDown( const Dst : TBitmap)
7796:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7797:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7798:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7799:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7800:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7801:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7802:     Procedure SmoothResize( var Src, Dst : TBitmap)
7803:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7804:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7805:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7806:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7807:     Procedure GrayScale( const Dst : TBitmap)
7808:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7809:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7810:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7811:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7812:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7813:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7814:     Procedure AntiAlias( const Dst : TBitmap)
7815:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7816:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7817:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7818:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7819:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7820:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7821:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7822:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7823:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7824:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7825:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7826:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7827:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7828:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7829:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7830:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7831:     Procedure Invert( Src : TBitmap)
7832:     Procedure MirrorRight( Src : TBitmap)
7833:     Procedure MirrorDown( Src : TBitmap)
7834:   end;
7835: end;
7836:
7837: (*----------------------------------------------------------------------------*)
7838: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7839: begin
7840:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7841:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7842:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7843:   SIRegister_TJvPaintFX(CL);
7844:  Function SplineFilter( Value : Single) : Single
7845:  Function BellFilter( Value : Single) : Single
7846:  Function TriangleFilter( Value : Single) : Single
7847:  Function BoxFilter( Value : Single) : Single
7848:  Function HermiteFilter( Value : Single) : Single
7849:  Function Lanczos3Filter( Value : Single) : Single
7850:  Function MitchellFilter( Value : Single) : Single
7851: end;
7852:
7853:
7854: (*----------------------------------------------------------------------------*)
```

```
7855: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7856: begin
7857:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7858:  TeeMsg_DefaultSeriesName','String 'Series
7859:  TeeMsg_DefaultToolName','String 'ChartTool
7860:  ChartComponentPalette','String 'TeeChart
7861:  TeeMaxLegendColumns',LongInt'( 2);
7862:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7863:  TeeTitleFootDistance,LongInt( 5);
7864:   SIRegister_TCustomChartWall(CL);
7865:   SIRegister_TChartWall(CL);
7866:   SIRegister_TChartLegendGradient(CL);
7867:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7868:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7869:   FindClass('TOBJECT'),'LegendException
7870:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7871:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7872:   FindClass('TOBJECT'),'TCustomChartLegend
7873:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7874:   TLegendSymbolPosition', '( spLeft, spRight )
7875:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7876:   TSymbolCalcHeight', 'Function  : Integer
7877:   SIRegister_TLegendSymbol(CL);
7878:   SIRegister_TTeeCustomShapePosition(CL);
7879:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7880:   SIRegister_TLegendTitle(CL);
7881:   SIRegister_TLegendItem(CL);
7882:   SIRegister_TLegendItems(CL);
7883:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7884:   FindClass('TOBJECT'),'TCustomChart
7885:   SIRegister_TCustomChartLegend(CL);
7886:   SIRegister_TChartLegend(CL);
7887:   SIRegister_TChartTitle(CL);
7888:   SIRegister_TChartFootTitle(CL);
7889:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7890:    +'eButton; Shift : TShiftState; X, Y : Integer)
7891:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7892:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7893:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7894:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7895:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7896:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7897:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7898:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7899:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7900:    +'toMax : Boolean; Min : Double; Max : Double; end
7901:   TAllAxisSavedScales', 'array of TAxisSavedScales
7902:   SIRegister_TChartBackWall(CL);
7903:   SIRegister_TChartRightWall(CL);
7904:   SIRegister_TChartBottomWall(CL);
7905:   SIRegister_TChartLeftWall(CL);
7906:   SIRegister_TChartWalls(CL);
7907:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7908:   SIRegister_TCustomChart(CL);
7909:   SIRegister_TChart(CL);
7910:   SIRegister_TTeeSeriesTypes(CL);
7911:   SIRegister_TTeeToolTypes(CL);
7912:   SIRegister_TTeeDragObject(CL);
7913:   SIRegister_TColorPalettes(CL);
7914:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7915:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7916:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7917:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7918:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7919:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7920:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7921:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7922:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7923:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7924:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7925:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7926:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7927:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7928:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7929:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7930:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7931:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7932:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7933:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7934:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7935:   SIRegister_TChartTheme(CL);
7936:   //TChartThemeClass', 'class of TChartTheme
7937:   //TCanvasClass', 'class of TCanvas3D
7938:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
```

```
7939:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7940:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7941:  Procedure ShowMessageUser( const S : String)
7942:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7943:  Function HasLabels( ASeries : TChartSeries) : Boolean
7944:  Function HasColors( ASeries : TChartSeries) : Boolean
7945:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7946:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7947: end;
7948:
7949:
7950: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7951: begin
7952:  //'TeeFormBorderStyle',' bsNone);
7953:   SIRegister_TMetafile(CL);
7954:  'TeeDefVerticalMargin','LongInt'( 4);
7955:  'TeeDefHorizMargin','LongInt'( 3);
7956:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7957:  'TeeMsg_TeeHand','String 'crTeeHand
7958:  'TeeNormalPrintDetail','LongInt'( 0);
7959:  'TeeHighPrintDetail','LongInt'( - 100);
7960:  'TeeDefault_PrintMargin','LongInt'( 15);
7961:  'MaxDefaultColors','LongInt'( 19);
7962:  'TeeTabDelimiter','Char #9);
7963:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7964:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7965:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7966:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7967:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7968:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7969:   SIRegister_TCustomPanelNoCaption(CL);
7970:   FindClass('TOBJECT'),'TCustomTeePanel
7971:   SIRegister_TZoomPanning(CL);
7972:   SIRegister_TTeeEvent(CL);
7973:   //SIRegister_TTeeEventListeners(CL);
7974:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7975:   SIRegister_TTeeMouseEvent(CL);
7976:   SIRegister_TCustomTeePanel(CL);
7977:   //TChartGradient', 'TTeeGradient
7978:   //TChartGradientClass', 'class of TChartGradient
7979:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7980:   SIRegister_TTeeZoomPen(CL);
7981:   SIRegister_TTeeZoomBrush(CL);
7982:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7983:   SIRegister_TTeeZoom(CL);
7984:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7985:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7986:   SIRegister_TBackImage(CL);
7987:   SIRegister_TCustomTeePanelExtended(CL);
7988:   //TChartBrushClass', 'class of TChartBrush
7989:   SIRegister_TTeeCustomShapeBrushPen(CL);
7990:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7991:   TTextFormat', '( ttfNormal, ttfHtml )
7992:   SIRegister_TTeeCustomShape(CL);
7993:   SIRegister_TTeeShape(CL);
7994:   SIRegister_TTeeExportData(CL);
7995:  Function TeeStr( const Num : Integer) : String
7996:  Function DateTimeDefaultFormat( const AStep : Double) : String
7997:  Function TEEDaysInMonth( Year, Month : Word) : Word
7998:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7999:  Function NextDateTimeStep( const AStep : Double) : Double
8000:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
8001:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
8002:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
8003:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
8004:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
8005:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
8006:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
8007:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
8008:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
8009:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
8010:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
8011:  Function DelphiToLocalFormat( const Format : String) : String
8012:  Function LocalToDelphiFormat( const Format : String) : String
8013:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8014:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8015:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8016:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
8017:   TTeeSortSwap', 'Procedure ( a, b : Integer)
8018:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8019:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
8020:  Function TeeExtractField( St : String; Index : Integer) : String;
8021:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
8022:  Function TeeNumFields( St : String) : Integer;
8023:  Function TeeNumFields1( const St, Separator : String) : Integer;
8024:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8025:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8026:  // TColorArray', 'array of TColor
```

```
8027:  Function GetDefaultColor( const Index : Integer) : TColor
8028:  Procedure SetDefaultColorPalette;
8029:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
8030:  'TeeCheckBoxSize','LongInt'( 11);
8031:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8032:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
8033:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8034:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8035:  Procedure TeeTranslateControl( AControl : TControl);
8036:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
8037:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8038:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8039:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8040:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8041:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8042:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8043:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8044:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8045:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8046:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8047:  Procedure TeeSaveStringOption( const AKey, Value : String)
8048:  Function TeeDefaultXMLEncoding : String
8049:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8050:   TeeWindowHandle', 'Integer
8051:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8052:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8053:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8054: end;
8055:
8056:
8057: using mXBDEUtils
8058: ********************************************************************************
8059:  Procedure SetAlias( aAlias, aDirectory : String)
8060:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
8061:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8062:  Procedure SetBDE( aPath, aNode, aValue : String)
8063:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8064:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8065:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8066:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8067:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8068:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8069:
8070:
8071: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8072: begin
8073: AddClassN(FindClass('TOBJECT'),'EDateTime
8074: Function DatePart( const D : TDateTime) : Integer
8075: Function TimePart( const D : TDateTime) : Double
8076: Function Century( const D : TDateTime) : Word
8077: Function Year( const D : TDateTime) : Word
8078: Function Month( const D : TDateTime) : Word
8079: Function Day( const D : TDateTime) : Word
8080: Function Hour( const D : TDateTime) : Word
8081: Function Minute( const D : TDateTime) : Word
8082: Function Second( const D : TDateTime) : Word
8083: Function Millisecond( const D : TDateTime) : Word
8084: ('OneDay','Extended').setExtended( 1.0);
8085: ('OneHour','Extended').SetExtended( OneDay / 24);
8086: ('OneMinute','Extended').SetExtended( OneHour / 60);
8087: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8088: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8089: ('OneWeek','Extended').SetExtended( OneDay * 7);
8090: ('HoursPerDay','Extended').SetExtended( 24);
8091: ('MinutesPerHour','Extended').SetExtended( 60);
8092: ('SecondsPerMinute','Extended').SetExtended( 60);
8093: Procedure SetYear( var D : TDateTime; const Year : Word)
8094: Procedure SetMonth( var D : TDateTime; const Month : Word)
8095: Procedure SetDay( var D : TDateTime; const Day : Word)
8096: Procedure SetHour( var D : TDateTime; const Hour : Word)
8097: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8098: Procedure SetSecond( var D : TDateTime; const Second : Word)
8099: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8100: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8101: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8102: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8103: Function IsAM( const D : TDateTime) : Boolean
8104: Function IsPM( const D : TDateTime) : Boolean
8105: Function IsMidnight( const D : TDateTime) : Boolean
8106: Function IsNoon( const D : TDateTime) : Boolean
8107: Function IsSunday( const D : TDateTime) : Boolean
8108: Function IsMonday( const D : TDateTime) : Boolean
8109: Function IsTuesday( const D : TDateTime) : Boolean
8110: Function IsWedneday( const D : TDateTime) : Boolean
8111: Function IsThursday( const D : TDateTime) : Boolean
8112: Function IsFriday( const D : TDateTime) : Boolean
8113: Function IsSaturday( const D : TDateTime) : Boolean
8114: Function IsWeekend( const D : TDateTime) : Boolean
```

```
8115: Function Noon( const D : TDateTime) : TDateTime
8116: Function Midnight( const D : TDateTime) : TDateTime
8117: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8118: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8119: Function NextWorkday( const D : TDateTime) : TDateTime
8120: Function PreviousWorkday( const D : TDateTime) : TDateTime
8121: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8122: Function LastDayOfYear( const D : TDateTime) : TDateTime
8123: Function EasterSunday( const Year : Word) : TDateTime
8124: Function GoodFriday( const Year : Word) : TDateTime
8125: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8126: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8127: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8128: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8129: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8130: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8131: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8132: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8133: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8134: Function DayOfYear( const D : TDateTime) : Integer
8135: Function DaysInMonth( const Ye, Mo : Word) : Integer
8136: Function DaysInMonth( const D : TDateTime) : Integer
8137: Function DaysInYear( const Ye : Word) : Integer
8138: Function DaysInYearDate( const D : TDateTime) : Integer
8139: Function WeekNumber( const D : TDateTime) : Integer
8140: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8141: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8142: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8143: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8144: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8145: Function DiffHours( const D1, D2 : TDateTime) : Integer
8146: Function DiffDays( const D1, D2 : TDateTime) : Integer
8147: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8148: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8149: Function DiffYears( const D1, D2 : TDateTime) : Integer
8150: Function GMTBias : Integer
8151: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8152: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8153: Function NowAsGMTTime : TDateTime
8154: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8155: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8156: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8157: Function DateTimeToANSI( const D : TDateTime) : Integer
8158: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8159: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8160: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8161: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8162: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8163: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8164: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8165: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8166: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8167: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8168: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8169: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8170: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8171: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8172: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8173: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8174: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8175: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8176: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8177: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8178: Function EnglishShortMonthA( const S : AnsiString) : Integer
8179: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8180: Function EnglishLongMonthA( const S : AnsiString) : Integer
8181: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8182: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8183: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8184: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8185: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8186: Function RFCMonthA( const S : AnsiString) : Word
8187: Function RFCMonthU( const S : UnicodeString) : Word
8188: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8189: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8190: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8191: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8192: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8193: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8194: Function NowAsRFCDateTimeA : AnsiString
8195: Function NowAsRFCDateTimeU : UnicodeString
8196: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8197: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8198: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8199: Function TimePeriodStr( const D : TDateTime) : AnsiString
8200: Procedure SelfTest
8201: end;
8202: //****************************************CFileUtils
8203: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
```

```
8204: Function PathHasDriveLetter( const Path : String) : Boolean
8205: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8206: Function PathIsDriveLetter( const Path : String) : Boolean
8207: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8208: Function PathIsDriveRoot( const Path : String) : Boolean
8209: Function PathIsRootA( const Path : AnsiString) : Boolean
8210: Function PathIsRoot( const Path : String) : Boolean
8211: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8212: Function PathIsUNCPath( const Path : String) : Boolean
8213: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8214: Function PathIsAbsolute( const Path : String) : Boolean
8215: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8216: Function PathIsDirectory( const Path : String) : Boolean
8217: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8218: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8219: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8220: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8221: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8222: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8223: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8224: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8225: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8226: Function PathCanonical( const Path : String; const PathSep : Char) : String
8227: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8228: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8229: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8230: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8231: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8232: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8233: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8234: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8235: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8236: Function FileNameValid( const FileName : String) : String
8237: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8238: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8239: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8240: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8241: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8242: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8243: Procedure CCopyFile( const FileName, DestName : String)
8244: Procedure CMoveFile( const FileName, DestName : String)
8245: Function CDeleteFiles( const FileMask : String) : Boolean
8246: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8247: Procedure FileCloseEx( const FileHandle : TFileHandle)
8248: Function FileExistsA( const FileName : AnsiString) : Boolean
8249:  Function CFileExists( const FileName : String) : Boolean
8250:  Function CFileGetSize( const FileName : String) : Int64
8251: Function FileGetDateTime( const FileName : String) : TDateTime
8252: Function FileGetDateTime2( const FileName : String) : TDateTime
8253: Function FileIsReadOnly( const FileName : String) : Boolean
8254: Procedure FileDeleteEx( const FileName : String)
8255: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8256: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8257: Function DirectoryEntryExists( const Name : String) : Boolean
8258: Function DirectoryEntrySize( const Name : String) : Int64
8259:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8260: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8261: Procedure CDirectoryCreate( const DirectoryName : String)
8262: Function GetFirstFileNameMatching( const FileMask : String) : String
8263: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8264: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8265: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8266:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8267:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8268: Function DriveIsValid( const Drive : Char) : Boolean
8269: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8270: Function DriveFreeSpace( const Path : AnsiString) : Int64
8271:
8272: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8273: begin
8274:  AddClassN(FindClass('TOBJECT'),'ETimers
8275:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8276: Function TickDelta( const D1, D2 : LongWord) : Integer
8277: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8278:   AddTypeS('THPTimer', 'Int64
8279: Procedure StartTimer( var Timer : THPTimer)
8280: Procedure StopTimer( var Timer : THPTimer)
8281: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8282: Procedure InitStoppedTimer( var Timer : THPTimer)
8283: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8284: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8285: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8286: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8287: Function GetHighPrecisionFrequency : Int64
8288: Function GetHighPrecisionTimerOverhead : Int64
8289: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8290: Procedure SelfTestCTimer
8291: end;
```

```
8292:
8293: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8294: begin
8295:  Function RandomSeed : LongWord
8296:  Procedure AddEntropy( const Value : LongWord)
8297:  Function RandomUniform : LongWord;
8298:  Function RandomUniform1( const N : Integer) : Integer;
8299:  Function RandomBoolean : Boolean
8300:  Function RandomByte : Byte
8301:  Function RandomByteNonZero : Byte
8302:  Function RandomWord : Word
8303:  Function RandomInt64 : Int64;
8304:  Function RandomInt641( const N : Int64) : Int64;
8305:  Function RandomHex( const Digits : Integer) : String
8306:  Function RandomFloat : Extended
8307:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8308:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8309:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8310:  Function mwcRandomLongWord : LongWord
8311:  Function urnRandomLongWord : LongWord
8312:  Function moaRandomFloat : Extended
8313:  Function mwcRandomFloat : Extended
8314:  Function RandomNormalF : Extended
8315:  Procedure SelfTestCRandom
8316: end;
8317:
8318: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8319: begin
8320:  // PIntArray', '^TIntArray // will not work
8321:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8322:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8323:  Function synMax( x, y : integer) : integer
8324:  Function synMin( x, y : integer) : integer
8325:  Function synMinMax( x, mi, ma : integer) : integer
8326:  Procedure synSwapInt( var l, r : integer)
8327:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8328:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8329:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8330:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8331:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8332:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8333:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8334:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8335:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8336:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8337:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8338:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8339:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8340:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8341:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8342:  ('C3_NONSPACING','LongInt'( 1);
8343:  'C3_DIACRITIC','LongInt'( 2);
8344:  'C3_VOWELMARK','LongInt'( 4);
8345:  ('C3_SYMBOL','LongInt'( 8);
8346:  ('C3_KATAKANA','LongWord( $0010);
8347:  ('C3_HIRAGANA','LongWord( $0020);
8348:  ('C3_HALFWIDTH','LongWord( $0040);
8349:  ('C3_FULLWIDTH','LongWord( $0080);
8350:  ('C3_IDEOGRAPH','LongWord( $0100);
8351:  ('C3_KASHIDA','LongWord( $0200);
8352:  ('C3_LEXICAL','LongWord( $0400);
8353:  ('C3_ALPHA','LongWord( $8000);
8354:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8355:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8356:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8357:  Function synIsStringType( Value : Word) : TStringType
8358:  Function synGetEOL( Line : PChar) : PChar
8359:  Function synEncodeString( s : string) : string
8360:  Function synDecodeString( s : string) : string
8361:  Procedure synFreeAndNil( var Obj: TObject)
8362:  Procedure synAssert( Expr : Boolean)
8363:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8364:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8365:   TReplaceFlags', 'set of TReplaceFlag )
8366:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8367:  Function synGetRValue( RGBValue : TColor) : byte
8368:  Function synGetGValue( RGBValue : TColor) : byte
8369:  Function synGetBValue( RGBValue : TColor) : byte
8370:  Function synRGB( r, g, b : Byte) : Cardinal
8371: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8372:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8373:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8374:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8375:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8376: end;
8377:
8378:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
```

```
8379:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8380:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8381:
8382: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8383: begin
8384:  Function STimeZoneBias : integer
8385:  Function TimeZone : string
8386:  Function Rfc822DateTime( t : TDateTime) : string
8387:  Function CDateTime( t : TDateTime) : string
8388:  Function SimpleDateTime( t : TDateTime) : string
8389:  Function AnsiCDateTime( t : TDateTime) : string
8390:  Function GetMonthNumber( Value : String) : integer
8391:  Function GetTimeFromStr( Value : string) : TDateTime
8392:  Function GetDateMDYFromStr( Value : string) : TDateTime
8393:  Function DecodeRfcDateTime( Value : string) : TDateTime
8394:  Function GetUTTime : TDateTime
8395:  Function SetUTTime( Newdt : TDateTime) : Boolean
8396:  Function SGetTick : LongWord
8397:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8398:  Function CodeInt( Value : Word) : Ansistring
8399:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8400:  Function CodeLongInt( Value : LongInt) : Ansistring
8401:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8402:  Function DumpStr( const Buffer : Ansistring) : string
8403:  Function DumpExStr( const Buffer : Ansistring) : string
8404:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8405:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8406:  Function TrimSPLeft( const S : string) : string
8407:  Function TrimSPRight( const S : string) : string
8408:  Function TrimSP( const S : string) : string
8409:  Function SeparateLeft( const Value, Delimiter : string) : string
8410:  Function SeparateRight( const Value, Delimiter : string) : string
8411:  Function SGetParameter( const Value, Parameter : string) : string
8412:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8413:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8414:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8415:  Function GetEmailAddr( const Value : string) : string
8416:  Function GetEmailDesc( Value : string) : string
8417:  Function CStrToHex( const Value : Ansistring) : string
8418:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8419:  Function CBinToInt( const Value : string) : Integer
8420:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8421:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8422:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8423:  Function CRPos( const Sub, Value : String) : Integer
8424:  Function FetchBin( var Value : string; const Delimiter : string) : string
8425:  Function CFetch( var Value : string; const Delimiter : string) : string
8426:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8427:  Function IsBinaryString( const Value : AnsiString) : Boolean
8428:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8429:  Procedure StringsTrim( const value : TStrings)
8430:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8431:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8432:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8433:  Function CCountOfChar( const Value : string; aChr : char) : integer
8434:  Function UnquoteStr( const Value : string; Quote : Char) : string
8435:  Function QuoteStr( const Value : string; Quote : Char) : string
8436:  Procedure HeadersToList( const Value : TStrings)
8437:  Procedure ListToHeaders( const Value : TStrings)
8438:  Function SwapBytes( Value : integer) : integer
8439:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8440:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8441:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8442:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8443:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8444:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8445: end;
8446:
8447: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8448: begin
8449:  ('CrcBufSize','LongInt'( 2048);
8450:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8451:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8452:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8453:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8454:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8455:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8456:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8457:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8458:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8459:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8460:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8461:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8462:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8463:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8464:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8465: end;
8466:
8467: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
```

```
8468: begin
8469:  function CreateOleObject(const ClassName: String): IDispatch;
8470:  function GetActiveOleObject(const ClassName: String): IDispatch;
8471:  function ProgIDToClassID(const ProgID: string): TGUID;
8472:  function ClassIDToProgID(const ClassID: TGUID): string;
8473:  function CreateClassID: string;
8474:  function CreateGUIDString: string;
8475:  function CreateGUIDID: string;
8476:  procedure OleError(ErrorCode: longint)
8477:  procedure OleCheck(Result: HResult);
8478: end;
8479:
8480:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8481:  Function xGetActiveOleObject( const ClassName : string) : Variant
8482:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8483:  Function DllCanUnloadNow : HResult
8484:  Function DllRegisterServer : HResult
8485:  Function DllUnregisterServer : HResult
8486:  Function VarFromInterface( Unknown : IUnknown) : Variant
8487:  Function VarToInterface( const V : Variant) : IDispatch
8488:  Function VarToAutoObject( const V : Variant) : TAutoObject
8489:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8490:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8491:  Procedure OleError( ErrorCode : HResult)
8492:  Procedure OleCheck( Result : HResult)
8493:  Function StringToClassID( const S : string) : TCLSID
8494:  Function ClassIDToString( const ClassID : TCLSID) : string
8495:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8496:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8497:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8498:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8499:  Function xGUIDToString( const ClassID : TGUID) : string
8500:  Function xStringToGUID( const S : string) : TGUID
8501:  Function xGetModuleName( Module : HMODULE) : string
8502:  Function xAcquireExceptionObject : TObject
8503:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8504:  Function xUtf8Encode( const WS : WideString) : UTF8String
8505:  Function xUtf8Decode( const S : UTF8String) : WideString
8506:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8507:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8508:  Function XRTLHandleCOMException : HResult
8509:  Procedure XRTLCheckArgument( Flag : Boolean)
8510:  //Procedure XRTLCheckOutArgument( out Arg)
8511:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8512:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8513:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8514:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8515:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8516:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8517: function   XRTLDefaultCategoryManager: IUnknown;
8518: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8519: // ICatRegister helper functions
8520: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8521:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8522:                                        const CategoryManager: IUnknown = nil): HResult;
8523: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8524:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8525:                                        const CategoryManager: IUnknown = nil): HResult;
8526: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8527:                                        const CategoryManager: IUnknown = nil): HResult;
8528: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8529:                                          const CategoryManager: IUnknown = nil): HResult;
8530: // ICatInformation helper functions
8531: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8532:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8533:                                       const CategoryManager: IUnknown = nil): HResult;
8534: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8535:                                const CategoryManager: IUnknown = nil): HResult;
8536: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8537:                                     const CategoryManager: IUnknown = nil): HResult;
8538: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8539:                                      const CategoryManager: IUnknown = nil): HResult;
8540: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8541:                    const ADelete: Boolean = True): WideString;
8542: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8543: Function XRTLGetVariantAsString( const Value : Variant) : string
8544: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8545: Function XRTLGetTimeZones : TXRTLTimeZones
8546: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8547: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8548: Function GMTNow : TDateTime
8549: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8550: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8551: Procedure XRTLNotImplemented
8552: Procedure XRTLRaiseError( E : Exception)
8553: Procedure XRTLRaise( E : Exception)');
```

```
8554: Procedure XRaise( E : Exception)');
8555: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8556:
8557:
8558: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8559: begin
8560:   SIRegister_IXRTLValue(CL);
8561:   SIRegister_TXRTLValue(CL);
8562:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8563:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8564: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8565: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8566: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8567: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8568: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8569: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8570: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8571: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8572: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8573: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8574: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8575: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8576: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8577: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8578: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8579: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8580: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8581: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8582: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8583: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8584: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8585: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8586: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8587: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8588: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8589: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8590: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8591:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8592: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8593: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8594: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8595: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8596: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8597: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8598: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8599: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8600: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8601: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8602: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8603: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8604: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8605: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8606: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8607: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8608: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8609: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8610: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8611: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8612: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8613: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8614: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8615: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8616: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8617: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8618: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8619: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8620: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8621: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8622: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8623: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8624: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8625: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8626: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8627: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8628: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8629: end;
8630:
8631: //****************************unit uPSI_GR32;*****************************************
8632:
8633:  Function Color32( WinColor : TColor) : TColor32;
8634:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8635:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8636:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8637:  Function WinColor( Color32 : TColor32) : TColor
8638:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8639:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8640:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8641:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
```

```
8642:  Function RedComponent( Color32 : TColor32) : Integer
8643:  Function GreenComponent( Color32 : TColor32) : Integer
8644:  Function BlueComponent( Color32 : TColor32) : Integer
8645:  Function AlphaComponent( Color32 : TColor32) : Integer
8646:  Function Intensity( Color32 : TColor32) : Integer
8647:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8648:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8649:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8650:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8651:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8652:  Function WinPalette( const P : TPalette32) : HPALETTE
8653:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8654:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8655:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8656:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8657:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8658:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8659:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8660:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8661:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8662:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8663:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8664:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8665:  Function FixedRect1( const ARect : TRect) : TRect;
8666:  Function FixedRect2( const FR : TFloatRect) : TRect;
8667:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8668:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8669:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8670:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8671:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8672:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8673:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8674:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8675:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8676:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8677:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8678:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8679:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8680:  Function IsRectEmpty( const R : TRect) : Boolean;
8681:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8682:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8683:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8684:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8685:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8686:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8687:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8688:  Function MessageBeep( uType : UINT) : BOOL
8689:  Function ShowCursor( bShow : BOOL) : Integer
8690:  Function SetCursorPos( X, Y : Integer) : BOOL
8691:  Function SetCursor( hCursor : HICON) : HCURSOR
8692:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8693:  //Function ClipCursor( lpRect : PRect) : BOOL
8694:  Function GetClipCursor( var lpRect : TRect) : BOOL
8695:  Function GetCursor : HCURSOR
8696:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8697:  Function GetCaretBlinkTime : UINT
8698:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8699:  Function DestroyCaret : BOOL
8700:  Function HideCaret( hWnd : HWND) : BOOL
8701:  Function ShowCaret( hWnd : HWND) : BOOL
8702:  Function SetCaretPos( X, Y : Integer) : BOOL
8703:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8704:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8705:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8706:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8707:  Function WindowFromPoint( Point : TPoint) : HWND
8708:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8709:
8710:
8711: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8712: begin
8713:  Function FixedFloor( A : TFixed) : Integer
8714:  Function FixedCeil( A : TFixed) : Integer
8715:  Function FixedMul( A, B : TFixed) : TFixed
8716:  Function FixedDiv( A, B : TFixed) : TFixed
8717:  Function OneOver( Value : TFixed) : TFixed
8718:  Function FixedRound( A : TFixed) : Integer
8719:  Function FixedSqr( Value : TFixed) : TFixed
8720:  Function FixedSqrtLP( Value : TFixed) : TFixed
8721:  Function FixedSqrtHP( Value : TFixed) : TFixed
8722:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8723:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8724:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8725:  Function GRHypot( const X, Y : TFloat) : TFloat;
8726:  Function Hypot1( const X, Y : Integer) : Integer;
8727:  Function FastSqrt( const Value : TFloat) : TFloat
8728:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8729:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8730:  Function FastInvSqrt( const Value : Single) : Single;
```

```
8731:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8732:  Function GRIsPowerOf2( Value : Integer) : Boolean
8733:  Function PrevPowerOf2( Value : Integer) : Integer
8734:  Function NextPowerOf2( Value : Integer) : Integer
8735:  Function Average( A, B : Integer) : Integer
8736:  Function GRSign( Value : Integer) : Integer
8737:  Function FloatMod( x, y : Double) : Double
8738: end;
8739:
8740: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8741: begin
8742:  Function Clamp( const Value : Integer) : Integer;
8743:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8744:  Function StackAlloc( Size : Integer) : Pointer
8745:  Procedure StackFree( P : Pointer)
8746:  Procedure Swap( var A, B : Pointer);
8747:  Procedure Swap1( var A, B : Integer);
8748:  Procedure Swap2( var A, B : TFixed);
8749:  Procedure Swap3( var A, B : TColor32);
8750:  Procedure TestSwap( var A, B : Integer);
8751:  Procedure TestSwap1( var A, B : TFixed);
8752:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8753:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8754:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8755:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8756:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8757:  Function GRMin( const A, B, C : Integer) : Integer;
8758:  Function GRMax( const A, B, C : Integer) : Integer;
8759:  Function Clamp( Value, Max : Integer) : Integer;
8760:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8761:  Function Wrap( Value, Max : Integer) : Integer;
8762:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8763:  Function Wrap3( Value, Max : Single) : Single;;
8764:  Function WrapPow2( Value, Max : Integer) : Integer;
8765:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8766:  Function Mirror( Value, Max : Integer) : Integer;
8767:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8768:  Function MirrorPow2( Value, Max : Integer) : Integer;
8769:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8770:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8771:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8772:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8773:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8774:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8775:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8776:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8777:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8778:  Function Div255( Value : Cardinal) : Cardinal
8779:  Function SAR_4( Value : Integer) : Integer
8780:  Function SAR_8( Value : Integer) : Integer
8781:  Function SAR_9( Value : Integer) : Integer
8782:  Function SAR_11( Value : Integer) : Integer
8783:  Function SAR_12( Value : Integer) : Integer
8784:  Function SAR_13( Value : Integer) : Integer
8785:  Function SAR_14( Value : Integer) : Integer
8786:  Function SAR_15( Value : Integer) : Integer
8787:  Function SAR_16( Value : Integer) : Integer
8788:  Function ColorSwap( WinColor : TColor) : TColor32
8789: end;
8790:
8791: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8792: begin
8793:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8794:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8795:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8796:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8797:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8798:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8799:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8800:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8801:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8802:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8803:  Function CreateBitmask( Components : TColor32Components) : TColor32
8804:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8805:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8806:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8807: end;
8808:
8809:
8810: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8811: begin
8812:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8813:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8814:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8815:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8816:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
```

```
8817:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8818:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8819:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8820:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8821:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8822:    //+'tedRangeBuffer; MoreData : Boolean; end
8823:  Function NtfsSetSparse( const FileName : string) : Boolean
8824:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8825:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8826:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8827:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8828:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8829:  Function NtfsGetSparse( const FileName : string) : Boolean
8830:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8831:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8832:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8833:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8834:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8835:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8836:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8837:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8838:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8839:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8840:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8841:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8842:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8843:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8844:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8845:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8846:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8847:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8848:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8849:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8850:   AddTypeS('TStreamIds', 'set of TStreamId
8851:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8852:    +': ___Pointer; StreamIds : TStreamIds; end
8853:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8854:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8855:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8856:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8857:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8858:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8859:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8860:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8861:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8862:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8863:  Function JclAppInstances : TJclAppInstances;
8864:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8865:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8866:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8867:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8868:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8869:
8870:
8871: (*----------------------------------------------------------------------------*)
8872: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8873: begin
8874:   FindClass('TOBJECT'),'EJclGraphicsError
8875:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8876:   TDynPointArray', 'array of TPoint
8877:   TDynDynPointArrayArray', 'array of TDynPointArray
8878:   TPointF', 'record X : Single; Y : Single; end
8879:   TDynPointArrayF', 'array of TPointF
8880:   TDrawMode2', '( dmOpaque, dmBlend )
8881:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8882:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8883:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8884:   TMatrix3d', 'record array[0..2,0..2] of extended end
8885:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8886:   TScanLine', 'array of Integer
8887:   TScanLines', 'array of TScanLine
8888:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8889:   TGradientDirection', '( gdVertical, gdHorizontal )
8890:   TPolyFillMode', '( fmAlternate, fmWinding )
8891:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8892:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8893:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8894:   SIRegister_TJclDesktopCanvas(CL);
8895:   FindClass('TOBJECT'),'TJclRegion
8896:   SIRegister_TJclRegionInfo(CL);
8897:   SIRegister_TJclRegion(CL);
8898:   SIRegister_TJclThreadPersistent(CL);
8899:   SIRegister_TJclCustomMap(CL);
8900:   SIRegister_TJclBitmap32(CL);
8901:   SIRegister_TJclByteMap(CL);
8902:   SIRegister_TJclTransformation(CL);
```

```
8903:   SIRegister_TJclLinearTransformation(CL);
8904:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8905:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8906:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8907:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8908:  Procedure BitmapToJPeg( const FileName : string)
8909:  Procedure JPegToBitmap( const FileName : string)
8910:  Function ExtractIconCount( const FileName : string) : Integer
8911:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8912:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8913:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8914:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8915:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8916:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8917:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8918:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8919:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8920:  Procedure ScreenShot1( bm : TBitmap);
8921:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8922:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8923:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8924:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8925:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8926:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8927:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8928:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8929:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8930:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8931:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8932:  Procedure Invert( Dst, Src : TJclBitmap32)
8933:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8934:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8935:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8936:  Procedure SetGamma( Gamma : Single)
8937: end;
8938:
8939: (*----------------------------------------------------------------------------*)
8940: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8941: begin
8942:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8943:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8944:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8945:  Function LockedDec( var Target : Integer) : Integer
8946:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8947:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8948:  Function LockedExchangeDec( var Target : Integer) : Integer
8949:  Function LockedExchangeInc( var Target : Integer) : Integer
8950:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8951:  Function LockedInc( var Target : Integer) : Integer
8952:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8953:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8954:   SIRegister_TJclDispatcherObject(CL);
8955:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8956:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8957:   SIRegister_TJclCriticalSection(CL);
8958:   SIRegister_TJclCriticalSectionEx(CL);
8959:   SIRegister_TJclEvent(CL);
8960:   SIRegister_TJclWaitableTimer(CL);
8961:   SIRegister_TJclSemaphore(CL);
8962:   SIRegister_TJclMutex(CL);
8963:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8964:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8965:   SIRegister_TJclOptex(CL);
8966:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8967:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8968:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8969:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8970:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8971:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8972:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8973:   PMeteredSection', '^TMeteredSection // will not work
8974:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8975:   SIRegister_TJclMeteredSection(CL);
8976:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8977:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8978:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8979:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8980:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8981:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8982:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8983:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8984:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
```

```
8985:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8986:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8987:   FindClass('TOBJECT'),'EJclCriticalSectionError
8988:   FindClass('TOBJECT'),'EJclEventError
8989:   FindClass('TOBJECT'),'EJclWaitableTimerError
8990:   FindClass('TOBJECT'),'EJclSemaphoreError
8991:   FindClass('TOBJECT'),'EJclMutexError
8992:   FindClass('TOBJECT'),'EJclMeteredSectionError
8993: end;
8994:
8995:
8996: //*************************unit uPSI_mORMotReport;
8997: Procedure SetCurrentPrinterAsDefault
8998: Function CurrentPrinterName : string
8999: Function mCurrentPrinterPaperSize : string
9000: Procedure UseDefaultPrinter
9001:
9002: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
9003: begin
9004:   with FindClass('TOBJECT'), 'TStream') do begin
9005:     IsAbstract := True;
9006:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
9007:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
9008:     function Read(Buffer:String;Count:LongInt):LongInt
9009:     function Write(Buffer:String;Count:LongInt):LongInt
9010:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
9011:     function WriteString(Buffer:String;Count:LongInt):LongInt
9012:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
9013:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
9014:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
9015:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
9016:
9017:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
9018:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
9019:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
9020:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
9021:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
9022:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
9023:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
9024:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
9025:
9026:     function Seek(Offset:LongInt;Origin:Word):LongInt
9027:     procedure ReadBuffer(Buffer:String;Count:LongInt)
9028:     procedure WriteBuffer(Buffer:String;Count:LongInt)
9029:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
9030:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
9031:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
9032:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
9033:
9034:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9035:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9036:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9037:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9038:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9039:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9040:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9041:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9042:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9043:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9044:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9045:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9046:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9047:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9048:
9049:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9050:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9051:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
9052:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
9053:     //READBUFFERAC
9054:     function InstanceSize: Longint
9055:     Procedure FixupResourceHeader( FixupInfo : Integer)
9056:     Procedure ReadResHeader
9057:
9058:     {$IFDEF DELPHI4UP}
9059:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9060:     {$ELSE}
9061:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9062:     {$ENDIF}
9063:     RegisterProperty('Position', 'LongInt', iptrw);
9064:     RegisterProperty('Size', 'LongInt', iptrw);
9065:   end;
9066: end;
9067:
9068:
9069: { ***************************************************************
9070:   Unit DMATH - Interface for DMATH.DLL
9071:    *************************************************************** }
9072: // see more docs/dmath_manual.pdf
9073:
```

```
9074: Function InitEval : Integer
9075: Procedure SetVariable( VarName : Char; Value : Float)
9076: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9077: Function Eval( ExpressionString : String) : Float
9078:
9079: unit dmath; //types are in built, others are external in DLL
9080: interface
9081: {$IFDEF DELPHI}
9082: uses
9083:   StdCtrls, Graphics;
9084: {$ENDIF}
9085: { ---------------------------------------------------------------
9086:   Types and constants
9087:   --------------------------------------------------------------- }
9088: {$i types.inc}
9089: { ---------------------------------------------------------------
9090:   Error handling
9091:   --------------------------------------------------------------- }
9092: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9093: { Sets the error code }
9094: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9095: { Sets error code and default function value }
9096: function MathErr : Integer; external 'dmath';
9097: { Returns the error code }
9098: { ---------------------------------------------------------------
9099:   Dynamic arrays
9100:   --------------------------------------------------------------- }
9101: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9102: { Sets the auto-initialization of arrays }
9103: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9104: { Creates floating point vector V[0..Ub] }
9105: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9106: { Creates integer vector V[0..Ub] }
9107: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9108: { Creates complex vector V[0..Ub] }
9109: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9110: { Creates boolean vector V[0..Ub] }
9111: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9112: { Creates string vector V[0..Ub] }
9113: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9114: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9115: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9116: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9117: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9118: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9119: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9120: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9121: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9122: { Creates string matrix A[0..Ub1, 0..Ub2] }
9123: { ---------------------------------------------------------------
9124:   Minimum, maximum, sign and exchange
9125:   --------------------------------------------------------------- }
9126: function FMin(X, Y : Float) : Float; external 'dmath';
9127: { Minimum of 2 reals }
9128: function FMax(X, Y : Float) : Float; external 'dmath';
9129: { Maximum of 2 reals }
9130: function IMin(X, Y : Integer) : Integer; external 'dmath';
9131: { Minimum of 2 integers }
9132: function IMax(X, Y : Integer) : Integer; external 'dmath';
9133: { Maximum of 2 integers }
9134: function Sgn(X : Float) : Integer; external 'dmath';
9135: { Sign (returns 1 if X = 0) }
9136: function Sgn0(X : Float) : Integer; external 'dmath';
9137: { Sign (returns 0 if X = 0) }
9138: function DSgn(A, B : Float) : Float; external 'dmath';
9139: { Sgn(B) * |A| }
9140: procedure FSwap(var X, Y : Float); external 'dmath';
9141: { Exchange 2 reals }
9142: procedure ISwap(var X, Y : Integer); external 'dmath';
9143: { Exchange 2 integers }
9144: { ---------------------------------------------------------------
9145:   Rounding functions
9146:   --------------------------------------------------------------- }
9147: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9148: { Rounds X to N decimal places }
9149: function Ceil(X : Float) : Integer; external 'dmath';
9150: { Ceiling function }
9151: function Floor(X : Float) : Integer; external 'dmath';
9152: { Floor function }
9153: { ---------------------------------------------------------------
9154:   Logarithms, exponentials and power
9155:   --------------------------------------------------------------- }
9156: function Expo(X : Float) : Float; external 'dmath';
9157: { Exponential }
9158: function Exp2(X : Float) : Float; external 'dmath';
9159: { 2^X }
9160: function Exp10(X : Float) : Float; external 'dmath';
9161: { 10^X }
9162: function Log(X : Float) : Float; external 'dmath';
```

```
9163: { Natural log }
9164: function Log2(X : Float) : Float; external 'dmath';
9165: { Log, base 2 }
9166: function Log10(X : Float) : Float; external 'dmath';
9167: { Decimal log }
9168: function LogA(X, A : Float) : Float; external 'dmath';
9169: { Log, base A }
9170: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9171: { X^N }
9172: function Power(X, Y : Float) : Float; external 'dmath';
9173: { X^Y, X >= 0 }
9174: { ---------------------------------------------------------------
9175:   Trigonometric functions
9176:   --------------------------------------------------------------- }
9177: function Pythag(X, Y : Float) : Float; external 'dmath';
9178: { Sqrt(X^2 + Y^2) }
9179: function FixAngle(Theta : Float) : Float; external 'dmath';
9180: { Set Theta in -Pi..Pi }
9181: function Tan(X : Float) : Float; external 'dmath';
9182: { Tangent }
9183: function ArcSin(X : Float) : Float; external 'dmath';
9184: { Arc sinus }
9185: function ArcCos(X : Float) : Float; external 'dmath';
9186: { Arc cosinus }
9187: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9188: { Angle (Ox, OM) with M(X,Y) }
9189: { ---------------------------------------------------------------
9190:   Hyperbolic functions
9191:   --------------------------------------------------------------- }
9192: function Sinh(X : Float) : Float; external 'dmath';
9193: { Hyperbolic sine }
9194: function Cosh(X : Float) : Float; external 'dmath';
9195: { Hyperbolic cosine }
9196: function Tanh(X : Float) : Float; external 'dmath';
9197: { Hyperbolic tangent }
9198: function ArcSinh(X : Float) : Float; external 'dmath';
9199: { Inverse hyperbolic sine }
9200: function ArcCosh(X : Float) : Float; external 'dmath';
9201: { Inverse hyperbolic cosine }
9202: function ArcTanh(X : Float) : Float; external 'dmath';
9203: { Inverse hyperbolic tangent }
9204: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9205: { Sinh & Cosh }
9206: { ---------------------------------------------------------------
9207:   Gamma function and related functions
9208:   --------------------------------------------------------------- }
9209: function Fact(N : Integer) : Float; external 'dmath';
9210: { Factorial }
9211: function SgnGamma(X : Float) : Integer; external 'dmath';
9212: { Sign of Gamma function }
9213: function Gamma(X : Float) : Float; external 'dmath';
9214: { Gamma function }
9215: function LnGamma(X : Float) : Float; external 'dmath';
9216: { Logarithm of Gamma function }
9217: function Stirling(X : Float) : Float; external 'dmath';
9218: { Stirling's formula for the Gamma function }
9219: function StirLog(X : Float) : Float; external 'dmath';
9220: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9221: function DiGamma(X : Float ) : Float; external 'dmath';
9222: { Digamma function }
9223: function TriGamma(X : Float ) : Float; external 'dmath';
9224: { Trigamma function }
9225: function IGamma(A, X : Float) : Float; external 'dmath';
9226: { Incomplete Gamma function}
9227: function JGamma(A, X : Float) : Float; external 'dmath';
9228: { Complement of incomplete Gamma function }
9229: function InvGamma(A, P : Float) : Float; external 'dmath';
9230: { Inverse of incomplete Gamma function }
9231: function Erf(X : Float) : Float; external 'dmath';
9232: { Error function }
9233: function Erfc(X : Float) : Float; external 'dmath';
9234: { Complement of error function }
9235: { ---------------------------------------------------------------
9236:   Beta function and related functions
9237:   --------------------------------------------------------------- }
9238: function Beta(X, Y : Float) : Float; external 'dmath';
9239: { Beta function }
9240: function IBeta(A, B, X : Float) : Float; external 'dmath';
9241: { Incomplete Beta function }
9242: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9243: { Inverse of incomplete Beta function }
9244: { ---------------------------------------------------------------
9245:   Lambert's function
9246:   --------------------------------------------------------------- }
9247: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9248:  ---------------------------------------------------------------
9249:   Binomial distribution
9250:   --------------------------------------------------------------- }
9251: function Binomial(N, K : Integer) : Float; external 'dmath';
```

```
9252: { Binomial coefficient C(N,K) }
9253: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9254: { Probability of binomial distribution }
9255: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9256: { Cumulative probability for binomial distrib. }
9257: { -----------------------------------------------------------------
9258:   Poisson distribution
9259:   ----------------------------------------------------------------- }
9260: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9261: { Probability of Poisson distribution }
9262: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9263: { Cumulative probability for Poisson distrib. }
9264: { -----------------------------------------------------------------
9265:   Exponential distribution
9266:   ----------------------------------------------------------------- }
9267: function DExpo(A, X : Float) : Float; external 'dmath';
9268: { Density of exponential distribution with parameter A }
9269: function FExpo(A, X : Float) : Float; external 'dmath';
9270: { Cumulative probability function for exponential dist. with parameter A }
9271: { -----------------------------------------------------------------
9272:   Standard normal distribution
9273:   ----------------------------------------------------------------- }
9274: function DNorm(X : Float) : Float; external 'dmath';
9275: { Density of standard normal distribution }
9276: function FNorm(X : Float) : Float; external 'dmath';
9277: { Cumulative probability for standard normal distrib. }
9278: function PNorm(X : Float) : Float; external 'dmath';
9279: { Prob(|U| > X) for standard normal distrib. }
9280: function InvNorm(P : Float) : Float; external 'dmath';
9281: { Inverse of standard normal distribution }
9282: { -----------------------------------------------------------------
9283:   Student's distribution
9284:   ----------------------------------------------------------------- }
9285: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9286: { Density of Student distribution with Nu d.o.f. }
9287: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9288: { Cumulative probability for Student distrib. with Nu d.o.f. }
9289: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9290: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9291: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9292: { Inverse of Student's t-distribution function }
9293: { -----------------------------------------------------------------
9294:   Khi-2 distribution
9295:   ----------------------------------------------------------------- }
9296: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9297: { Density of Khi-2 distribution with Nu d.o.f. }
9298: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9299: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9300: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9301: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9302: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9303: { Inverse of Khi-2 distribution function }
9304: { -----------------------------------------------------------------
9305:   Fisher-Snedecor distribution
9306:   ----------------------------------------------------------------- }
9307: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9308: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9309: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9310: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9311: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9312: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9313: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9314: { Inverse of Snedecor's F-distribution function }
9315: { -----------------------------------------------------------------
9316:   Beta distribution
9317:   ----------------------------------------------------------------- }
9318: function DBeta(A, B, X : Float) : Float; external 'dmath';
9319: { Density of Beta distribution with parameters A and B }
9320: function FBeta(A, B, X : Float) : Float; external 'dmath';
9321: { Cumulative probability for Beta distrib. with param. A and B }
9322: { -----------------------------------------------------------------
9323:   Gamma distribution
9324:   ----------------------------------------------------------------- }
9325: function DGamma(A, B, X : Float) : Float; external 'dmath';
9326: { Density of Gamma distribution with parameters A and B }
9327: function FGamma(A, B, X : Float) : Float; external 'dmath';
9328: { Cumulative probability for Gamma distrib. with param. A and B }
9329: { -----------------------------------------------------------------
9330:   Expression evaluation
9331:   ----------------------------------------------------------------- }
9332: function InitEval : Integer; external 'dmath';
9333: { Initializes built-in functions and returns their number }
9334: function Eval(ExpressionString : String) : Float; external 'dmath';
9335: { Evaluates an expression at run-time }
9336: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9337: { Assigns a value to a variable }
9338: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9339: { Adds a function to the parser }
9340: { -----------------------------------------------------------------
```

```
9341:   Matrices and linear equations
9342:   ------------------------------------------------------------------ }
9343: procedure GaussJordan(A           : TMatrix;
9344:                       Lb, Ub1, Ub2 : Integer;
9345:                       var Det      : Float); external 'dmath';
9346: { Transforms a matrix according to the Gauss-Jordan method }
9347: procedure LinEq(A       : TMatrix;
9348:                 B       : TVector;
9349:                 Lb, Ub  : Integer;
9350:                 var Det : Float); external 'dmath';
9351: { Solves a linear system according to the Gauss-Jordan method }
9352: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9353: { Cholesky factorization of a positive definite symmetric matrix }
9354: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9355: { LU decomposition }
9356: procedure LU_Solve(A      : TMatrix;
9357:                    B      : TVector;
9358:                    Lb, Ub : Integer;
9359:                    X      : TVector); external 'dmath';
9360: { Solution of linear system from LU decomposition }
9361: procedure QR_Decomp(A           : TMatrix;
9362:                     Lb, Ub1, Ub2 : Integer;
9363:                     R           : TMatrix); external 'dmath';
9364: { QR decomposition }
9365: procedure QR_Solve(Q, R         : TMatrix;
9366:                    B           : TVector;
9367:                    Lb, Ub1, Ub2 : Integer;
9368:                    X           : TVector); external 'dmath';
9369: { Solution of linear system from QR decomposition }
9370: procedure SV_Decomp(A           : TMatrix;
9371:                     Lb, Ub1, Ub2 : Integer;
9372:                     S           : TVector;
9373:                     V           : TMatrix); external 'dmath';
9374: { Singular value decomposition }
9375: procedure SV_SetZero(S      : TVector;
9376:                      Lb, Ub : Integer;
9377:                      Tol    : Float); external 'dmath';
9378: { Set lowest singular values to zero }
9379: procedure SV_Solve(U           : TMatrix;
9380:                    S           : TVector;
9381:                    V           : TMatrix;
9382:                    B           : TVector;
9383:                    Lb, Ub1, Ub2 : Integer;
9384:                    X           : TVector); external 'dmath';
9385: { Solution of linear system from SVD }
9386: procedure SV_Approx(U           : TMatrix;
9387:                     S           : TVector;
9388:                     V           : TMatrix;
9389:                     Lb, Ub1, Ub2 : Integer;
9390:                     A           : TMatrix); external 'dmath';
9391: { Matrix approximation from SVD }
9392: procedure EigenVals(A      : TMatrix;
9393:                     Lb, Ub : Integer;
9394:                     Lambda : TCompVector); external 'dmath';
9395: { Eigenvalues of a general square matrix }
9396: procedure EigenVect(A      : TMatrix;
9397:                     Lb, Ub : Integer;
9398:                     Lambda : TCompVector;
9399:                     V      : TMatrix); external 'dmath';
9400: { Eigenvalues and eigenvectors of a general square matrix }
9401: procedure EigenSym(A      : TMatrix;
9402:                    Lb, Ub : Integer;
9403:                    Lambda : TVector;
9404:                    V      : TMatrix); external 'dmath';
9405: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9406: procedure Jacobi(A              : TMatrix;
9407:                  Lb, Ub, MaxIter : Integer;
9408:                  Tol            : Float;
9409:                  Lambda         : TVector;
9410:                  V              : TMatrix); external 'dmath';
9411: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9412: { ------------------------------------------------------------------
9413:   Optimization
9414:   ------------------------------------------------------------------ }
9415: procedure MinBrack(Func                    : TFunc;
9416:                    var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9417: { Brackets a minimum of a function }
9418: procedure GoldSearch(Func          : TFunc;
9419:                      A, B          : Float;
9420:                      MaxIter       : Integer;
9421:                      Tol           : Float;
9422:                      var Xmin, Ymin : Float); external 'dmath';
9423: { Minimization of a function of one variable (golden search) }
9424: procedure LinMin(Func      : TFuncNVar;
9425:                  X, DeltaX : TVector;
9426:                  Lb, Ub    : Integer;
9427:                  var R     : Float;
9428:                  MaxIter   : Integer;
9429:                  Tol       : Float;
```

```
9430:                 var F_min : Float); external 'dmath';
9431: { Minimization of a function of several variables along a line }
9432: procedure Newton(Func      : TFuncNVar;
9433:                  HessGrad  : THessGrad;
9434:                  X         : TVector;
9435:                  Lb, Ub    : Integer;
9436:                  MaxIter   : Integer;
9437:                  Tol       : Float;
9438:                  var F_min : Float;
9439:                  G         : TVector;
9440:                  H_inv     : TMatrix;
9441:                  var Det   : Float); external 'dmath';
9442: { Minimization of a function of several variables (Newton's method) }
9443: procedure SaveNewton(FileName : string); external 'dmath';
9444: { Save Newton iterations in a file }
9445: procedure Marquardt(Func      : TFuncNVar;
9446:                     HessGrad  : THessGrad;
9447:                     X         : TVector;
9448:                     Lb, Ub    : Integer;
9449:                     MaxIter   : Integer;
9450:                     Tol       : Float;
9451:                     var F_min : Float;
9452:                     G         : TVector;
9453:                     H_inv     : TMatrix;
9454:                     var Det   : Float); external 'dmath';
9455: { Minimization of a function of several variables (Marquardt's method) }
9456: procedure SaveMarquardt(FileName : string); external 'dmath';
9457: { Save Marquardt iterations in a file }
9458: procedure BFGS(Func      : TFuncNVar;
9459:                Gradient  : TGradient;
9460:                X         : TVector;
9461:                Lb, Ub    : Integer;
9462:                MaxIter   : Integer;
9463:                Tol       : Float;
9464:                var F_min : Float;
9465:                G         : TVector;
9466:                H_inv     : TMatrix); external 'dmath';
9467: { Minimization of a function of several variables (BFGS method) }
9468: procedure SaveBFGS(FileName : string); external 'dmath';
9469: { Save BFGS iterations in a file }
9470: procedure Simplex(Func      : TFuncNVar;
9471:                   X         : TVector;
9472:                   Lb, Ub    : Integer;
9473:                   MaxIter   : Integer;
9474:                   Tol       : Float;
9475:                   var F_min : Float); external 'dmath';
9476: { Minimization of a function of several variables (Simplex) }
9477: procedure SaveSimplex(FileName : string); external 'dmath';
9478: { Save Simplex iterations in a file }
9479: { ------------------------------------------------------------------
9480:   Nonlinear equations
9481:   ------------------------------------------------------------------ }
9482: procedure RootBrack(Func             : TFunc;
9483:                     var X, Y, FX, FY : Float); external 'dmath';
9484: { Brackets a root of function Func between X and Y }
9485: procedure Bisect(Func     : TFunc;
9486:                  var X, Y : Float;
9487:                  MaxIter  : Integer;
9488:                  Tol      : Float;
9489:                  var F    : Float); external 'dmath';
9490: { Bisection method }
9491: procedure Secant(Func     : TFunc;
9492:                  var X, Y : Float;
9493:                  MaxIter  : Integer;
9494:                  Tol      : Float;
9495:                  var F    : Float); external 'dmath';
9496: { Secant method }
9497: procedure NewtEq(Func, Deriv : TFunc;
9498:                  var X       : Float;
9499:                  MaxIter     : Integer;
9500:                  Tol         : Float;
9501:                  var F       : Float); external 'dmath';
9502: { Newton-Raphson method for a single nonlinear equation }
9503: procedure NewtEqs(Equations : TEquations;
9504:                   Jacobian  : TJacobian;
9505:                   X, F      : TVector;
9506:                   Lb, Ub    : Integer;
9507:                   MaxIter   : Integer;
9508:                   Tol       : Float); external 'dmath';
9509: { Newton-Raphson method for a system of nonlinear equations }
9510: procedure Broyden(Equations : TEquations;
9511:                   X, F      : TVector;
9512:                   Lb, Ub    : Integer;
9513:                   MaxIter   : Integer;
9514:                   Tol       : Float); external 'dmath';
9515: { Broyden's method for a system of nonlinear equations }
9516: { ------------------------------------------------------------------
9517:   Polynomials and rational fractions
9518:   ------------------------------------------------------------------ }
```

```
9519: function Poly(X    : Float;
9520:               Coef : TVector;
9521:               Deg  : Integer) : Float; external 'dmath';
9522: { Evaluates a polynomial }
9523: function RFrac(X          : Float;
9524:                Coef       : TVector;
9525:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9526: { Evaluates a rational fraction }
9527: function RootPol1(A, B  : Float;
9528:                   var X : Float) : Integer; external 'dmath';
9529: { Solves the linear equation A + B * X = 0 }
9530: function RootPol2(Coef : TVector;
9531:                   Z    : TCompVector) : Integer; external 'dmath';
9532: { Solves a quadratic equation }
9533: function RootPol3(Coef : TVector;
9534:                   Z    : TCompVector) : Integer; external 'dmath';
9535: { Solves a cubic equation }
9536: function RootPol4(Coef : TVector;
9537:                   Z    : TCompVector) : Integer; external 'dmath';
9538: { Solves a quartic equation }
9539: function RootPol(Coef : TVector;
9540:                  Deg  : Integer;
9541:                  Z    : TCompVector) : Integer; external 'dmath';
9542: { Solves a polynomial equation }
9543: function SetRealRoots(Deg : Integer;
9544:                       Z   : TCompVector;
9545:                       Tol : Float) : Integer; external 'dmath';
9546: { Set the imaginary part of a root to zero }
9547: procedure SortRoots(Deg : Integer;
9548:                     Z   : TCompVector); external 'dmath';
9549: { Sorts the roots of a polynomial }
9550: { ------------------------------------------------------------------
9551:   Numerical integration and differential equations
9552:   ------------------------------------------------------------------ }
9553: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9554: { Integration by trapezoidal rule }
9555: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9556: { Integral from A to B }
9557: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9558: { Integral from 0 to B }
9559: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9560: { Convolution product at time T }
9561: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9562: { Convolution by trapezoidal rule }
9563: procedure RKF45(F                    : TDiffEqs;
9564:                 Neqn                 : Integer;
9565:                 Y, Yp                : TVector;
9566:                 var T                : Float;
9567:                 Tout, RelErr, AbsErr : Float;
9568:                 var Flag             : Integer); external 'dmath';
9569: { Integration of a system of differential equations }
9570: { ------------------------------------------------------------------
9571:   Fast Fourier Transform
9572:   ------------------------------------------------------------------ }
9573: procedure FFT(NumSamples        : Integer;
9574:               InArray, OutArray : TCompVector); external 'dmath';
9575: { Fast Fourier Transform }
9576: procedure IFFT(NumSamples        : Integer;
9577:                InArray, OutArray : TCompVector); external 'dmath';
9578: { Inverse Fast Fourier Transform }
9579: procedure FFT_Integer(NumSamples     : Integer;
9580:                       RealIn, ImagIn : TIntVector;
9581:                       OutArray       : TCompVector); external 'dmath';
9582: { Fast Fourier Transform for integer data }
9583: procedure FFT_Integer_Cleanup; external 'dmath';
9584: { Clear memory after a call to FFT_Integer }
9585: procedure CalcFrequency(NumSamples,
9586:                         FrequencyIndex : Integer;
9587:                         InArray        : TCompVector;
9588:                         var FFT        : Complex); external 'dmath';
9589: { Direct computation of Fourier transform }
9590: { ------------------------------------------------------------------
9591:   Random numbers
9592:   ------------------------------------------------------------------ }
9593: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9594: { Select generator }
9595: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9596: { Initialize generator }
9597: function IRanGen : RNG_IntType; external 'dmath';
9598: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9599: function IRanGen31 : RNG_IntType; external 'dmath';
9600: { 31-bit random integer in [0 .. 2^31 - 1] }
9601: function RanGen1 : Float; external 'dmath';
9602: { 32-bit random real in [0,1] }
9603: function RanGen2 : Float; external 'dmath';
9604: { 32-bit random real in [0,1) }
9605: function RanGen3 : Float; external 'dmath';
9606: { 32-bit random real in (0,1) }
9607: function RanGen53 : Float; external 'dmath';
```

```
9608: { 53-bit random real in [0,1) }
9609: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9610: { Initializes the 'Multiply with carry' random number generator }
9611: function IRanMWC : RNG_IntType; external 'dmath';
9612: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9613: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9614: { Initializes Mersenne Twister generator with a seed }
9615: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9616:                         KeyLength : Word); external 'dmath';
9617: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9618: function IRanMT : RNG_IntType; external 'dmath';
9619: { Random integer from MT generator }
9620: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9621: { Initializes the UVAG generator with a string }
9622: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9623: { Initializes the UVAG generator with an integer }
9624: function IRanUVAG : RNG_IntType; external 'dmath';
9625: { Random integer from UVAG generator }
9626: function RanGaussStd : Float; external 'dmath';
9627: { Random number from standard normal distribution }
9628: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9629: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9630: procedure RanMult(M      : TVector; L      : TMatrix;
9631:                   Lb, Ub : Integer;
9632:                   X      : TVector); external 'dmath';
9633: { Random vector from multinormal distribution (correlated) }
9634: procedure RanMultIndep(M, S   : TVector;
9635:                        Lb, Ub : Integer;
9636:                        X      : TVector); external 'dmath';
9637: { Random vector from multinormal distribution (uncorrelated) }
9638: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9639: { Initializes Metropolis-Hastings parameters }
9640: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9641: { Returns Metropolis-Hastings parameters }
9642: procedure Hastings(Func      : TFuncNVar;
9643:                    T         : Float;
9644:                    X         : TVector;
9645:                    V         : TMatrix;
9646:                    Lb, Ub    : Integer;
9647:                    Xmat      : TMatrix;
9648:                    X_min     : TVector;
9649:                    var F_min : Float); external 'dmath';
9650: { Simulation of a probability density function by Metropolis-Hastings }
9651: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9652: { Initializes Simulated Annealing parameters }
9653: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9654: { Initializes log file }
9655: procedure SimAnn(Func          : TFuncNVar;
9656:                  X, Xmin, Xmax : TVector;
9657:                  Lb, Ub        : Integer;
9658:                  var F_min     : Float); external 'dmath';
9659: { Minimization of a function of several var. by simulated annealing }
9660: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9661: { Initializes Genetic Algorithm parameters }
9662: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9663: { Initializes log file }
9664: procedure GenAlg(Func          : TFuncNVar;
9665:                  X, Xmin, Xmax : TVector;
9666:                  Lb, Ub        : Integer;
9667:                  var F_min     : Float); external 'dmath';
9668: { Minimization of a function of several var. by genetic algorithm }
9669: { ------------------------------------------------------------------
9670:   Statistics
9671:   ------------------------------------------------------------------ }
9672: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9673: { Mean of sample X }
9674: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9675: { Minimum of sample X }
9676: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9677: { Maximum of sample X }
9678: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9679: { Median of sample X }
9680: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9681: { Standard deviation estimated from sample X }
9682: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9683: { Standard deviation of population }
9684: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9685: { Correlation coefficient }
9686: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9687: { Skewness of sample X }
9688: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9689: { Kurtosis of sample X }
9690: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9691: { Quick sort (ascending order) }
9692: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9693: { Quick sort (descending order) }
9694: procedure Interval(X1, X2             : Float;
9695:                    MinDiv, MaxDiv     : Integer;
9696:                    var Min, Max, Step : Float); external 'dmath';
```

```
9697: { Determines an interval for a set of values }
9698: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9699:                     var XMin, XMax, XStep : Float); external 'dmath';
9700: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9701: procedure StudIndep(N1, N2          : Integer;
9702:                     M1, M2, S1, S2 : Float;
9703:                     var T          : Float;
9704:                     var DoF        : Integer); external 'dmath';
9705: { Student t-test for independent samples }
9706: procedure StudPaired(X, Y    : TVector;
9707:                      Lb, Ub  : Integer;
9708:                      var T   : Float;
9709:                      var DoF : Integer); external 'dmath';
9710: { Student t-test for paired samples }
9711: procedure AnOVa1(Ns               : Integer;
9712:                  N                : TIntVector;
9713:                  M, S             : TVector;
9714:                  var V_f, V_r, F  : Float;
9715:                  var DoF_f, DoF_r : Integer); external 'dmath';
9716: { One-way analysis of variance }
9717: procedure AnOVa2(NA, NB, Nobs : Integer;
9718:                  M, S         : TMatrix;
9719:                  V, F         : TVector;
9720:                  DoF          : TIntVector); external 'dmath';
9721: { Two-way analysis of variance }
9722: procedure Snedecor(N1, N2         : Integer;
9723:                    S1, S2         : Float;
9724:                    var F          : Float;
9725:                    var DoF1, DoF2 : Integer); external 'dmath';
9726: { Snedecor's F-test (comparison of two variances) }
9727: procedure Bartlett(Ns       : Integer;
9728:                    N        : TIntVector;
9729:                    S        : TVector;
9730:                    var Khi2 : Float;
9731:                    var DoF  : Integer); external 'dmath';
9732: { Bartlett's test (comparison of several variances) }
9733: procedure Mann_Whitney(N1, N2     : Integer;
9734:                        X1, X2     : TVector;
9735:                        var U, Eps : Float); external 'dmath';
9736: { Mann-Whitney test}
9737: procedure Wilcoxon(X, Y       : TVector;
9738:                    Lb, Ub     : Integer;
9739:                    var Ndiff  : Integer;
9740:                    var T, Eps : Float); external 'dmath';
9741: { Wilcoxon test }
9742: procedure Kruskal_Wallis(Ns      : Integer;
9743:                          N       : TIntVector;
9744:                          X       : TMatrix;
9745:                          var H   : Float;
9746:                          var DoF : Integer); external 'dmath';
9747: { Kruskal-Wallis test }
9748: procedure Khi2_Conform(N_cls    : Integer;
9749:                        N_estim  : Integer;
9750:                        Obs      : TIntVector;
9751:                        Calc     : TVector;
9752:                        var Khi2 : Float;
9753:                        var DoF  : Integer); external 'dmath';
9754: { Khi-2 test for conformity }
9755: procedure Khi2_Indep(N_lin    : Integer;
9756:                      N_col    : Integer;
9757:                      Obs      : TIntMatrix;
9758:                      var Khi2 : Float;
9759:                      var DoF  : Integer); external 'dmath';
9760: { Khi-2 test for independence }
9761: procedure Woolf_Conform(N_cls   : Integer;
9762:                         N_estim : Integer;
9763:                         Obs     : TIntVector;
9764:                         Calc    : TVector;
9765:                         var G   : Float;
9766:                         var DoF : Integer); external 'dmath';
9767: { Woolf's test for conformity }
9768: procedure Woolf_Indep(N_lin   : Integer;
9769:                       N_col   : Integer;
9770:                       Obs     : TIntMatrix;
9771:                       var G   : Float;
9772:                       var DoF : Integer); external 'dmath';
9773: { Woolf's test for independence }
9774: procedure DimStatClassVector(var C : TStatClassVector;
9775:                              Ub    : Integer); external 'dmath';
9776: { Allocates an array of statistical classes: C[0..Ub] }
9777: procedure Distrib(X       : TVector;
9778:                   Lb, Ub  : Integer;
9779:                   A, B, H : Float;
9780:                   C       : TStatClassVector); external 'dmath';
9781: { Distributes an array X[Lb..Ub] into statistical classes }
9782: { ---------------------------------------------------------------
9783:   Linear / polynomial regression
9784:   --------------------------------------------------------------- }
9785: procedure LinFit(X, Y   : TVector;
```

```
9786:                 Lb, Ub : Integer;
9787:                 B      : TVector;
9788:                 V      : TMatrix); external 'dmath';
9789: { Linear regression : Y = B(0) + B(1) * X }
9790: procedure WLinFit(X, Y, S : TVector;
9791:                   Lb, Ub  : Integer;
9792:                   B       : TVector;
9793:                   V       : TMatrix); external 'dmath';
9794: { Weighted linear regression : Y = B(0) + B(1) * X }
9795: procedure SVDLinFit(X, Y   : TVector;
9796:                     Lb, Ub : Integer;
9797:                     SVDTol : Float;
9798:                     B      : TVector;
9799:                     V      : TMatrix); external 'dmath';
9800: { Unweighted linear regression by singular value decomposition }
9801: procedure WSVDLinFit(X, Y, S : TVector;
9802:                      Lb, Ub  : Integer;
9803:                      SVDTol  : Float;
9804:                      B       : TVector;
9805:                      V       : TMatrix); external 'dmath';
9806: { Weighted linear regression by singular value decomposition }
9807: procedure MulFit(X            : TMatrix;
9808:                  Y            : TVector;
9809:                  Lb, Ub, Nvar : Integer;
9810:                  ConsTerm     : Boolean;
9811:                  B            : TVector;
9812:                  V            : TMatrix); external 'dmath';
9813: { Multiple linear regression by Gauss-Jordan method }
9814: procedure WMulFit(X            : TMatrix;
9815:                   Y, S         : TVector;
9816:                   Lb, Ub, Nvar : Integer;
9817:                   ConsTerm     : Boolean;
9818:                   B            : TVector;
9819:                   V            : TMatrix); external 'dmath';
9820: { Weighted multiple linear regression by Gauss-Jordan method }
9821: procedure SVDFit(X            : TMatrix;
9822:                  Y            : TVector;
9823:                  Lb, Ub, Nvar : Integer;
9824:                  ConsTerm     : Boolean;
9825:                  SVDTol       : Float;
9826:                  B            : TVector;
9827:                  V            : TMatrix); external 'dmath';
9828: { Multiple linear regression by singular value decomposition }
9829: procedure WSVDFit(X            : TMatrix;
9830:                   Y, S         : TVector;
9831:                   Lb, Ub, Nvar : Integer;
9832:                   ConsTerm     : Boolean;
9833:                   SVDTol       : Float;
9834:                   B            : TVector;
9835:                   V            : TMatrix); external 'dmath';
9836: { Weighted multiple linear regression by singular value decomposition }
9837: procedure PolFit(X, Y        : TVector;
9838:                  Lb, Ub, Deg : Integer;
9839:                  B           : TVector;
9840:                  V           : TMatrix); external 'dmath';
9841: { Polynomial regression by Gauss-Jordan method }
9842: procedure WPolFit(X, Y, S     : TVector;
9843:                   Lb, Ub, Deg : Integer;
9844:                   B           : TVector;
9845:                   V           : TMatrix); external 'dmath';
9846: { Weighted polynomial regression by Gauss-Jordan method }
9847: procedure SVDPolFit(X, Y        : TVector;
9848:                     Lb, Ub, Deg : Integer;
9849:                     SVDTol      : Float;
9850:                     B           : TVector;
9851:                     V           : TMatrix); external 'dmath';
9852: { Unweighted polynomial regression by singular value decomposition }
9853: procedure WSVDPolFit(X, Y, S     : TVector;
9854:                      Lb, Ub, Deg : Integer;
9855:                      SVDTol      : Float;
9856:                      B           : TVector;
9857:                      V           : TMatrix); external 'dmath';
9858: { Weighted polynomial regression by singular value decomposition }
9859: procedure RegTest(Y, Ycalc : TVector;
9860:                   LbY, UbY : Integer;
9861:                   V        : TMatrix;
9862:                   LbV, UbV : Integer;
9863:                   var Test : TRegTest); external 'dmath';
9864: { Test of unweighted regression }
9865: procedure WRegTest(Y, Ycalc, S : TVector;
9866:                    LbY, UbY    : Integer;
9867:                    V           : TMatrix;
9868:                    LbV, UbV    : Integer;
9869:                    var Test    : TRegTest); external 'dmath';
9870: { Test of weighted regression }
9871: { ---------------------------------------------------------------
9872:   Nonlinear regression
9873:   --------------------------------------------------------------- }
9874: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
```

```
9875: { Sets the optimization algorithm for nonlinear regression }
9876: function GetOptAlgo : TOptAlgo; external 'dmath';
9877: { Returns the optimization algorithm }
9878: procedure SetMaxParam(N : Byte); external 'dmath';
9879: { Sets the maximum number of regression parameters for nonlinear regression }
9880: function GetMaxParam : Byte; external 'dmath';
9881: { Returns the maximum number of regression parameters for nonlinear regression }
9882: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9883: { Sets the bounds on the I-th regression parameter }
9884: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9885: { Returns the bounds on the I-th regression parameter }
9886: procedure NLFit(RegFunc   : TRegFunc;
9887:                 DerivProc : TDerivProc;
9888:                 X, Y      : TVector;
9889:                 Lb, Ub    : Integer;
9890:                 MaxIter   : Integer;
9891:                 Tol       : Float;
9892:                 B         : TVector;
9893:                 FirstPar,
9894:                 LastPar   : Integer;
9895:                 V         : TMatrix); external 'dmath';
9896: { Unweighted nonlinear regression }
9897: procedure WNLFit(RegFunc   : TRegFunc;
9898:                  DerivProc : TDerivProc;
9899:                  X, Y, S   : TVector;
9900:                  Lb, Ub    : Integer;
9901:                  MaxIter   : Integer;
9902:                  Tol       : Float;
9903:                  B         : TVector;
9904:                  FirstPar,
9905:                  LastPar   : Integer;
9906:                  V         : TMatrix); external 'dmath';
9907: { Weighted nonlinear regression }
9908: procedure SetMCFile(FileName : String); external 'dmath';
9909: { Set file for saving MCMC simulations }
9910: procedure SimFit(RegFunc   : TRegFunc;
9911:                  X, Y      : TVector;
9912:                  Lb, Ub    : Integer;
9913:                  B         : TVector;
9914:                  FirstPar,
9915:                  LastPar   : Integer;
9916:                  V         : TMatrix); external 'dmath';
9917: { Simulation of unweighted nonlinear regression by MCMC }
9918: procedure WSimFit(RegFunc   : TRegFunc;
9919:                   X, Y, S   : TVector;
9920:                   Lb, Ub    : Integer;
9921:                   B         : TVector;
9922:                   FirstPar,
9923:                   LastPar   : Integer;
9924:                   V         : TMatrix); external 'dmath';
9925: { Simulation of weighted nonlinear regression by MCMC }
9926: { ------------------------------------------------------------------
9927:   Nonlinear regression models
9928:   ------------------------------------------------------------------ }
9929: procedure FracFit(X, Y       : TVector;
9930:                   Lb, Ub     : Integer;
9931:                   Deg1, Deg2 : Integer;
9932:                   ConsTerm   : Boolean;
9933:                   MaxIter    : Integer;
9934:                   Tol        : Float;
9935:                   B          : TVector;
9936:                   V          : TMatrix); external 'dmath';
9937: { Unweighted fit of rational fraction }
9938: procedure WFracFit(X, Y, S    : TVector;
9939:                    Lb, Ub     : Integer;
9940:                    Deg1, Deg2 : Integer;
9941:                    ConsTerm   : Boolean;
9942:                    MaxIter    : Integer;
9943:                    Tol        : Float;
9944:                    B          : TVector;
9945:                    V          : TMatrix); external 'dmath';
9946: { Weighted fit of rational fraction }
9947:
9948: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9949: { Returns the value of the rational fraction at point X }
9950: procedure ExpFit(X, Y         : TVector;
9951:                  Lb, Ub, Nexp : Integer;
9952:                  ConsTerm     : Boolean;
9953:                  MaxIter      : Integer;
9954:                  Tol          : Float;
9955:                  B            : TVector;
9956:                  V            : TMatrix); external 'dmath';
9957: { Unweighted fit of sum of exponentials }
9958: procedure WExpFit(X, Y, S      : TVector;
9959:                   Lb, Ub, Nexp : Integer;
9960:                   ConsTerm     : Boolean;
9961:                   MaxIter      : Integer;
9962:                   Tol          : Float;
9963:                   B            : TVector;
```

```
 9964:                  V           : TMatrix); external 'dmath';
 9965: { Weighted fit of sum of exponentials }
 9966: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9967: { Returns the value of the regression function at point X }
 9968: procedure IncExpFit(X, Y     : TVector;
 9969:                     Lb, Ub   : Integer;
 9970:                     ConsTerm : Boolean;
 9971:                     MaxIter  : Integer;
 9972:                     Tol      : Float;
 9973:                     B        : TVector;
 9974:                     V        : TMatrix); external 'dmath';
 9975: { Unweighted fit of model of increasing exponential }
 9976: procedure WIncExpFit(X, Y, S  : TVector;
 9977:                      Lb, Ub   : Integer;
 9978:                      ConsTerm : Boolean;
 9979:                      MaxIter  : Integer;
 9980:                      Tol      : Float;
 9981:                      B        : TVector;
 9982:                      V        : TMatrix); external 'dmath';
 9983: { Weighted fit of increasing exponential }
 9984: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9985: { Returns the value of the regression function at point X }
 9986: procedure ExpLinFit(X, Y    : TVector;
 9987:                     Lb, Ub  : Integer;
 9988:                     MaxIter : Integer;
 9989:                     Tol     : Float;
 9990:                     B       : TVector;
 9991:                     V       : TMatrix); external 'dmath';
 9992: { Unweighted fit of the "exponential + linear" model }
 9993: procedure WExpLinFit(X, Y, S : TVector;
 9994:                      Lb, Ub  : Integer;
 9995:                      MaxIter : Integer;
 9996:                      Tol     : Float;
 9997:                      B       : TVector;
 9998:                      V       : TMatrix); external 'dmath';
 9999: { Weighted fit of the "exponential + linear" model }
10000:
10001: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10002: { Returns the value of the regression function at point X }
10003: procedure MichFit(X, Y    : TVector;
10004:                   Lb, Ub  : Integer;
10005:                   MaxIter : Integer;
10006:                   Tol     : Float;
10007:                   B       : TVector;
10008:                   V       : TMatrix); external 'dmath';
10009: { Unweighted fit of Michaelis equation }
10010: procedure WMichFit(X, Y, S : TVector;
10011:                    Lb, Ub  : Integer;
10012:                    MaxIter : Integer;
10013:                    Tol     : Float;
10014:                    B       : TVector;
10015:                    V       : TMatrix); external 'dmath';
10016: { Weighted fit of Michaelis equation }
10017: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10018: { Returns the value of the Michaelis equation at point X }
10019: procedure MintFit(X, Y    : TVector;
10020:                   Lb, Ub  : Integer;
10021:                   MintVar : TMintVar;
10022:                   Fit_S0  : Boolean;
10023:                   MaxIter : Integer;
10024:                   Tol     : Float;
10025:                   B       : TVector;
10026:                   V       : TMatrix); external 'dmath';
10027: { Unweighted fit of the integrated Michaelis equation }
10028: procedure WMintFit(X, Y, S : TVector;
10029:                    Lb, Ub  : Integer;
10030:                    MintVar : TMintVar;
10031:                    Fit_S0  : Boolean;
10032:                    MaxIter : Integer;
10033:                    Tol     : Float;
10034:                    B       : TVector;
10035:                    V       : TMatrix); external 'dmath';
10036: { Weighted fit of the integrated Michaelis equation }
10037: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10038: { Returns the value of the integrated Michaelis equation at point X }
10039: procedure HillFit(X, Y    : TVector;
10040:                   Lb, Ub  : Integer;
10041:                   MaxIter : Integer;
10042:                   Tol     : Float;
10043:                   B       : TVector;
10044:                   V       : TMatrix); external 'dmath';
10045: { Unweighted fit of Hill equation }
10046: procedure WHillFit(X, Y, S : TVector;
10047:                    Lb, Ub  : Integer;
10048:                    MaxIter : Integer;
10049:                    Tol     : Float;
10050:                    B       : TVector;
10051:                    V       : TMatrix); external 'dmath';
10052: { Weighted fit of Hill equation }
```

```
10053: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10054: { Returns the value of the Hill equation at point X }
10055: procedure LogiFit(X, Y     : TVector;
10056:                   Lb, Ub   : Integer;
10057:                   ConsTerm : Boolean;
10058:                   General  : Boolean;
10059:                   MaxIter  : Integer;
10060:                   Tol      : Float;
10061:                   B        : TVector;
10062:                   V        : TMatrix); external 'dmath';
10063: { Unweighted fit of logistic function }
10064: procedure WLogiFit(X, Y, S  : TVector;
10065:                    Lb, Ub   : Integer;
10066:                    ConsTerm : Boolean;
10067:                    General  : Boolean;
10068:                    MaxIter  : Integer;
10069:                    Tol      : Float;
10070:                    B        : TVector;
10071:                    V        : TMatrix); external 'dmath';
10072: { Weighted fit of logistic function }
10073: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10074: { Returns the value of the logistic function at point X }
10075: procedure PKFit(X, Y    : TVector;
10076:                 Lb, Ub  : Integer;
10077:                 MaxIter : Integer;
10078:                 Tol     : Float;
10079:                 B       : TVector;
10080:                 V       : TMatrix); external 'dmath';
10081: { Unweighted fit of the acid-base titration curve }
10082: procedure WPKFit(X, Y, S : TVector;
10083:                  Lb, Ub  : Integer;
10084:                  MaxIter : Integer;
10085:                  Tol     : Float;
10086:                  B       : TVector;
10087:                  V       : TMatrix); external 'dmath';
10088: { Weighted fit of the acid-base titration curve }
10089: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10090: { Returns the value of the acid-base titration function at point X }
10091: procedure PowFit(X, Y    : TVector;
10092:                  Lb, Ub  : Integer;
10093:                  MaxIter : Integer;
10094:                  Tol     : Float;
10095:                  B       : TVector;
10096:                  V       : TMatrix); external 'dmath';
10097: { Unweighted fit of power function }
10098: procedure WPowFit(X, Y, S : TVector;
10099:                   Lb, Ub  : Integer;
10100:                   MaxIter : Integer;
10101:                   Tol     : Float;
10102:                   B       : TVector;
10103:                   V       : TMatrix); external 'dmath';
10104: { Weighted fit of power function }
10105:
10106: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10107: { Returns the value of the power function at point X }
10108: procedure GammaFit(X, Y    : TVector;
10109:                    Lb, Ub  : Integer;
10110:                    MaxIter : Integer;
10111:                    Tol     : Float;
10112:                    B       : TVector;
10113:                    V       : TMatrix); external 'dmath';
10114: { Unweighted fit of gamma distribution function }
10115: procedure WGammaFit(X, Y, S : TVector;
10116:                     Lb, Ub  : Integer;
10117:                     MaxIter : Integer;
10118:                     Tol     : Float;
10119:                     B       : TVector;
10120:                     V       : TMatrix); external 'dmath';
10121: { Weighted fit of gamma distribution function }
10122: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10123: { Returns the value of the gamma distribution function at point X }
10124: { ---------------------------------------------------------------
10125:   Principal component analysis
10126:   --------------------------------------------------------------- }
10127: procedure VecMean(X            : TMatrix;
10128:                   Lb, Ub, Nvar : Integer;
10129:                   M            : TVector); external 'dmath';
10130: { Computes the mean vector M from matrix X }
10131: procedure VecSD(X            : TMatrix;
10132:                 Lb, Ub, Nvar : Integer;
10133:                 M, S         : TVector); external 'dmath';
10134: { Computes the vector of standard deviations S from matrix X }
10135: procedure MatVarCov(X            : TMatrix;
10136:                     Lb, Ub, Nvar : Integer;
10137:                     M            : TVector;
10138:                     V            : TMatrix); external 'dmath';
10139: { Computes the variance-covariance matrix V from matrix X }
10140: procedure MatCorrel(V    : TMatrix;
10141:                     Nvar : Integer;
```

```
10142:                     R    : TMatrix); external 'dmath';
10143: { Computes the correlation matrix R from the var-cov matrix V }
10144: procedure PCA(R      : TMatrix;
10145:               Nvar   : Integer;
10146:               Lambda : TVector;
10147:               C, Rc  : TMatrix); external 'dmath';
10148: { Performs a principal component analysis of the correlation matrix R }
10149: procedure ScaleVar(X           : TMatrix;
10150:                    Lb, Ub, Nvar : Integer;
10151:                    M, S         : TVector;
10152:                    Z            : TMatrix); external 'dmath';
10153: { Scales a set of variables by subtracting means and dividing by SD's }
10154: procedure PrinFac(Z           : TMatrix;
10155:                   Lb, Ub, Nvar : Integer;
10156:                   C, F         : TMatrix); external 'dmath';
10157: { Computes principal factors }
10158: { ---------------------------------------------------------------
10159:   Strings
10160:   --------------------------------------------------------------- }
10161: function LTrim(S : String) : String; external 'dmath';
10162: { Removes leading blanks }
10163: function RTrim(S : String) : String; external 'dmath';
10164: { Removes trailing blanks }
10165: function Trim(S : String) : String; external 'dmath';
10166: { Removes leading and trailing blanks }
10167: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10168: { Returns a string made of character C repeated N times }
10169: function RFill(S : String; L : Byte) : String; external 'dmath';
10170: { Completes string S with trailing blanks for a total length L }
10171: function LFill(S : String; L : Byte) : String; external 'dmath';
10172: { Completes string S with leading blanks for a total length L }
10173: function CFill(S : String; L : Byte) : String; external 'dmath';
10174: { Centers string S on a total length L }
10175: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10176: { Replaces in string S all the occurences of C1 by C2 }
10177: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10178: { Extracts a field from a string }
10179: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10180: { Parses a string into its constitutive fields }
10181: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10182: { Sets the numeric format }
10183: function FloatStr(X : Float) : String; external 'dmath';
10184: { Converts a real to a string according to the numeric format }
10185: function IntStr(N : LongInt) : String; external 'dmath';
10186: { Converts an integer to a string }
10187: function CompStr(Z : Complex) : String; external 'dmath';
10188: { Converts a complex number to a string }
10189: {$IFDEF DELPHI}
10190: function StrDec(S : String) : String; external 'dmath';
10191: { Set decimal separator to the symbol defined in SysUtils }
10192: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10193: { Test if a string represents a number and returns it in X }
10194: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10195: { Reads a floating point number from an Edit control }
10196: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10197: { Writes a floating point number in a text file }
10198: {$ENDIF}
10199: { ---------------------------------------------------------------
10200:   BGI / Delphi graphics
10201:   --------------------------------------------------------------- }
10202: function InitGraphics
10203: {$IFDEF DELPHI}
10204: (Width, Height : Integer) : Boolean;
10205: {$ELSE}
10206: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10207: { Enters graphic mode }
10208: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10209:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10210: { Sets the graphic window }
10211: procedure SetOxScale(Scale               : TScale;
10212:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10213: { Sets the scale on the Ox axis }
10214: procedure SetOyScale(Scale               : TScale;
10215:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10216: { Sets the scale on the Oy axis }
10217: procedure GetOxScale(var Scale               : TScale;
10218:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10219: { Returns the scale on the Ox axis }
10220: procedure GetOyScale(var Scale               : TScale;
10221:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10222: { Returns the scale on the Oy axis }
10223: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10224: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10225: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10226: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10227: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10228: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10229: {$IFNDEF DELPHI}
10230: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
```

```
10231: { Sets the font for the main graph title }
10232: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10233: { Sets the font for the Ox axis (title and labels) }
10234: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10235: { Sets the font for the Oy axis (title and labels) }
10236: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10237: { Sets the font for the legends }
10238: procedure SetClipping(Clip : Boolean); external 'dmath';
10239: { Determines whether drawings are clipped at the current viewport
10240:   boundaries, according to the value of the Boolean parameter Clip }
10241: {$ENDIF}
10242: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10243: { Plots the horizontal axis }
10244: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10245: { Plots the vertical axis }
10246: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10247: { Plots a grid on the graph }
10248: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10249: { Writes the title of the graph }
10250: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10251: { Sets the maximum number of curves and re-initializes their parameters }
10252: procedure SetPointParam
10253: {$IFDEF DELPHI}
10254: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10255: {$ELSE}
10256: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10257: { Sets the point parameters for curve # CurvIndex }
10258: procedure SetLineParam
10259: {$IFDEF DELPHI}
10260: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10261: {$ELSE}
10262: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10263: { Sets the line parameters for curve # CurvIndex }
10264: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10265: { Sets the legend for curve # CurvIndex }
10266: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10267: { Sets the step for curve # CurvIndex }
10268: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10269: procedure GetPointParam
10270: {$IFDEF DELPHI}
10271: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10272: {$ELSE}
10273: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10274: { Returns the point parameters for curve # CurvIndex }
10275: procedure GetLineParam
10276: {$IFDEF DELPHI}
10277: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10278: {$ELSE}
10279: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10280: { Returns the line parameters for curve # CurvIndex }
10281: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10282: { Returns the legend for curve # CurvIndex }
10283: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10284: { Returns the step for curve # CurvIndex }
10285: {$IFDEF DELPHI}
10286: procedure PlotPoint(Canvas    : TCanvas;
10287:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10288: {$ELSE}
10289: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10290: {$ENDIF}
10291: { Plots a point on the screen }
10292: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10293:                     X, Y                 : TVector;
10294:                     Lb, Ub, CurvIndex    : Integer); external 'dmath';
10295: { Plots a curve }
10296: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10297:                                  X, Y, S             : TVector;
10298:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10299: { Plots a curve with error bars }
10300: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10301:                    Func                 : TFunc;
10302:                    Xmin, Xmax           : Float;
10303:                    {$IFDEF DELPHI}Npt   : Integer;{$ENDIF}
10304:                    CurvIndex            : Integer); external 'dmath';
10305: { Plots a function }
10306: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10307:                       NCurv             : Integer;
10308:                       ShowPoints, ShowLines : Boolean); external 'dmath';
10309: { Writes the legends for the plotted curves }
10310: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10311:                  Nx, Ny, Nc           : Integer;
10312:                  X, Y, Z              : TVector;
10313:                  F                    : TMatrix); external 'dmath';
10314: { Contour plot }
10315: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10316: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10317: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10318: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10319: {$IFNDEF DELPHI}
```

```
10320: procedure LeaveGraphics; external 'dmath';
10321: { Quits graphic mode }
10322: {$ENDIF}
10323: { ------------------------------------------------------------------
10324:   LaTeX graphics
10325:   ------------------------------------------------------------------ }
10326: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10327:                           Header            : Boolean) : Boolean; external 'dmath';
10328: { Initializes the LaTeX file }
10329: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10330: { Sets the graphic window }
10331: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10332: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10333: { Sets the scale on the Ox axis }
10334: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10335: { Sets the scale on the Oy axis }
10336: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10337: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10338: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10339: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10340: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10341: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10342: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10343: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10344: { Sets the maximum number of curves and re-initializes their parameters }
10345: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10346: { Sets the point parameters for curve # CurvIndex }
10347: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10348:                            Width : Float; Smooth : Boolean); external 'dmath';
10349: { Sets the line parameters for curve # CurvIndex }
10350: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10351: { Sets the legend for curve # CurvIndex }
10352: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10353: { Sets the step for curve # CurvIndex }
10354: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10355: { Plots a curve }
10356: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10357:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10358: { Plots a curve with error bars }
10359: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10360:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10361: { Plots a function }
10362: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10363: { Writes the legends for the plotted curves }
10364: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10365: { Contour plot }
10366: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10367: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10368:
10369: //******************************************************unit uPSI_SynPdf;
10370:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10371:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10372:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10373:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10374:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10375:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10376:  //Function _GetCharCount( Text : PAnsiChar) : integer
10377:  //Procedure L2R( W : PWideChar; L : integer)
10378:  Function PdfCoord( MM : single) : integer
10379:  Function CurrentPrinterPaperSize : TPDFPaperSize
10380:  Function CurrentPrinterRes : TPoint
10381:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10382:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10383:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10384:  Const('Usp10','String 'usp10.dll
10385:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10386:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10387:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10388: //*********************************************************************
10389:
10390: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10391: begin
10392:  Procedure PMrandomize( I : word)
10393:  Function PMrandom : longint
10394:  Function Rrand : extended
10395:  Function Irand( N : word) : word
10396:  Function Brand( P : extended) : boolean
10397:  Function Nrand : extended
10398: end;
10399:
10400: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10401: begin
10402:   Function Endian( x : LongWord) : LongWord
10403:   Function Endian64( x : Int64) : Int64
10404:   Function spRol( x : LongWord; y : Byte) : LongWord
10405:   Function spRor( x : LongWord; y : Byte) : LongWord
10406:   Function Ror64( x : Int64; y : Byte) : Int64
10407: end;
10408:
```

```
10409: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10410: begin
10411:  Procedure ClearModules
10412:  Procedure ReadMapFile( Fname : string)
10413:  Function AddressInfo( Address : dword) : string
10414: end;
10415:
10416: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10417: begin
10418:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10419:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10420:    +'teByOther, tpExecuteByOther )
10421:   TTarPermissions', 'set of TTarPermission
10422:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10423:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10424:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10425:   TTarModes', 'set of TTarMode
10426:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10427:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10428:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10429:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10430:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10431:   SIRegister_TTarArchive(CL);
10432:   SIRegister_TTarWriter(CL);
10433:  Function PermissionString( Permissions : TTarPermissions) : STRING
10434:  Function ConvertFilename( Filename : STRING) : STRING
10435:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10436:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10437:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10438: end;
10439:
10440:
10441: //************************************************unit uPSI_TlHelp32;
10442: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10443: begin
10444:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10445:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10446:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10447:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10448:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10449:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10450:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10451:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10452:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10453:   AddTypeS('THeapList32', 'tagHEAPLIST32
10454:  Const('HF32_DEFAULT','LongInt'( 1);
10455:  Const('HF32_SHARED','LongInt'( 2);
10456:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10457:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10458:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10459:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10460:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10461:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10462:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10463:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10464:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10465:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10466:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10467:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10468:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10469:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10470:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10471:    +'aPri : Longint; dwFlags : DWORD; end
10472:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10473:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10474:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10475:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10476: end;
10477:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10478:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10479:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10480:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10481:  Const('EWX_LOGOFF','LongInt'( 0);
10482:  Const('EWX_SHUTDOWN','LongInt'( 1);
10483:  Const('EWX_REBOOT','LongInt'( 2);
10484:  Const('EWX_FORCE','LongInt'( 4);
10485:  Const('EWX_POWEROFF','LongInt'( 8);
10486:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10487:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10488:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10489:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10490:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10491:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10492:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10493:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10494:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10495:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10496:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10497:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
```

```
10498:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10499:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10500:  Function GetDesktopWindow : HWND
10501:  Function GetParent( hWnd : HWND) : HWND
10502:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10503:  Function GetTopWindow( hWnd : HWND) : HWND
10504:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10505:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10506:  //Delphi DFM
10507:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10508:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10509:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10510:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10511:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10512:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10513:  Function OpenIcon( hWnd : HWND) : BOOL
10514:  Function CloseWindow( hWnd : HWND) : BOOL
10515:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10516:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10517:  Function IsWindowVisible( hWnd : HWND) : BOOL
10518:  Function IsIconic( hWnd : HWND) : BOOL
10519:  Function AnyPopup : BOOL
10520:  Function BringWindowToTop( hWnd : HWND) : BOOL
10521:  Function IsZoomed( hWnd : HWND) : BOOL
10522:  Function IsWindow( hWnd : HWND) : BOOL
10523:  Function IsMenu( hMenu : HMENU) : BOOL
10524:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10525:  Function DestroyWindow( hWnd : HWND) : BOOL
10526:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10527:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10528:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10529:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10530:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10531:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10532:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10533:
10534: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10535: begin
10536:  const('ShowSetupDialogOptLong','String '--setup
10537:  PrimaryConfPathOptLong','String '--primary-config-path=
10538:  PrimaryConfPathOptShort','String '--pcp=
10539:  SecondaryConfPathOptLong','String '--secondary-config-path=
10540:  SecondaryConfPathOptShort','String '--scp=
10541:  NoSplashScreenOptLong','String '--no-splash-screen
10542:  NoSplashScreenOptShort','String '--nsc
10543:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10544:  SkipLastProjectOpt','String '--skip-last-project
10545:  DebugLogOpt','String '--debug-log=
10546:  DebugLogOptEnable','String '--debug-enable=
10547:  LanguageOpt','String '--language=
10548:  LazarusDirOpt','String '--lazarusdir=
10549:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10550:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10551:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10552:  Function IsHelpRequested : Boolean
10553:  Function IsVersionRequested : boolean
10554:  Function GetLanguageSpecified : string
10555:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10556:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10557:  Procedure ParseNoGuiCmdLineParams
10558:  Function ExtractCmdLineFilenames : TStrings
10559: end;
10560:
10561:
10562: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10563: begin
10564:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10565:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10566:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10567:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10568:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10569:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10570:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10571:  Function DirPathExists( DirectoryName : string) : boolean
10572:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10573:  Function ExtractFileNameOnly( const AFilename : string) : string
10574:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10575:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10576:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10577:  Function ForceDirectory( DirectoryName : string) : boolean
10578:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10579:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10580:  Function FileIsText( const AFilename : string) : boolean
10581:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10582:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10583:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10584:  Function TrimFilename( const AFilename : string) : string
10585:  Function ResolveDots( const AFilename : string) : string
10586:  Procedure ForcePathDelims( var FileName : string)
```

```
10587:  Function GetForcedPathDelims( const FileName : string) : String
10588:  Function CleanAndExpandFilename( const Filename : string) : string
10589:  Function CleanAndExpandDirectory( const Filename : string) : string
10590:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10591:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10592:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10593:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10594:  Function FileIsInPath( const Filename, Path : string) : boolean
10595:  Function AppendPathDelim( const Path : string) : string
10596:  Function ChompPathDelim( const Path : string) : string
10597:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10598:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10599:  Function MinimizeSearchPath( const SearchPath : string) : string
10600:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10601:  (*Function FileExistsUTF8( const Filename : string) : boolean
10602:  Function FileAgeUTF8( const FileName : string) : Longint
10603:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10604:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10605:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10606:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10607:  Procedure FindCloseUTF8( var F : TSearchrec)
10608:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10609:  Function FileGetAttrUTF8( const FileName : String) : Longint
10610:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10611:  Function DeleteFileUTF8( const FileName : String) : Boolean
10612:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10613:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10614:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10615:  Function GetCurrentDirUTF8 : String
10616:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10617:  Function CreateDirUTF8( const NewDir : String) : Boolean
10618:  Function RemoveDirUTF8( const Dir : String) : Boolean
10619:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10620:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10621:  Function FileCreateUTF8( const FileName : string) : THandle;
10622:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10623:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10624:  Function FileSizeUtf8( const Filename : string) : int64
10625:  Function GetFileDescription( const AFilename : string) : string
10626:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10627:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10628:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10629:  Function IsUNCPath( const Path : String) : Boolean
10630:  Function ExtractUNCVolume( const Path : String) : String
10631:  Function ExtractFileRoot( FileName : String) : String
10632:  Function GetDarwinSystemFilename( Filename : string) : string
10633:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10634:  Function StrToCmdLineParam( const Param : string) : string
10635:  Function MergeCmdLineParams( ParamList : TStrings) : string
10636:  Procedure InvalidateFileStateCache( const Filename : string)
10637:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10638:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10639:  Function FindAllDocs(const Root, extmask: string): TStringlist;
10640:  Function ReadFileToString( const Filename : string) : string
10641:  procedure Inc1(var X: longint; N: Longint);
10642:
10643:  type
10644:   TCopyFileFlag = ( cffOverwriteFile,
10645:     cffCreateDestDirectory, cffPreserveTime );
10646:   TCopyFileFlags = set of TCopyFileFlag;*)
10647:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10648:   TCopyFileFlags', 'set of TCopyFileFlag
10649:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10650: end;
10651:
10652: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10653: begin
10654:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10655:   SIRegister_TMask(CL);
10656:   SIRegister_TParseStringList(CL);
10657:   SIRegister_TMaskList(CL);
10658:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10659:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10660:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10661:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10662: end;
10663:
10664: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10665: begin
10666:   //PShellHookInfo', '^TShellHookInfo // will not work
10667:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10668:   SHELLHOOKINFO', 'TShellHookInfo
10669:   LPSHELLHOOKINFO', 'PShellHookInfo
10670:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10671:   SIRegister_TJvShellHook(CL);
10672:  Function InitJvShellHooks : Boolean
10673:  Procedure UnInitJvShellHooks
```

```
10674: end;
10675:
10676: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10677: begin
10678:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10679:    +', dcHasSetSel, dcWantTab, dcNative )
10680:   TDlgCodes', 'set of TDlgCode
10681:  'dcWantMessage',' dcWantAllKeys);
10682:   SIRegister_IJvExControl(CL);
10683:   SIRegister_IJvDenySubClassing(CL);
10684:   SIRegister_TStructPtrMessage(CL);
10685:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10686:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10687:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10688:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10689:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10690:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10691:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10692:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10693:  Function GetFocusedControl( AControl : TControl) : TWinControl
10694:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10695:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10696:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10697:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10698:   SIRegister_TJvExControl(CL);
10699:   SIRegister_TJvExWinControl(CL);
10700:   SIRegister_TJvExCustomControl(CL);
10701:   SIRegister_TJvExGraphicControl(CL);
10702:   SIRegister_TJvExHintWindow(CL);
10703:   SIRegister_TJvExPubGraphicControl(CL);
10704: end;
10705:
10706: (*----------------------------------------------------------------------------*)
10707: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10708: begin
10709:  Procedure EncodeStream( Input, Output : TStream)
10710:  Procedure DecodeStream( Input, Output : TStream)
10711:  Function EncodeString1( const Input : string) : string
10712:  Function DecodeString1( const Input : string) : string
10713: end;
10714:
10715: (*----------------------------------------------------------------------------*)
10716: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10717: begin
10718:   SIRegister_TWebAppRegInfo(CL);
10719:   SIRegister_TWebAppRegList(CL);
10720:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10721:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10722:  Procedure UnregisterWebApp( const AProgID : string)
10723:  Function FindRegisteredWebApp( const AProgID : string) : string
10724:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10725:  'sUDPPort','String 'UDPPort
10726: end;
10727:
10728: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10729: begin
10730:  // TStringDynArray', 'array of string
10731:  Function GetEnvVarValue( const VarName : string) : string
10732:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10733:  Function DeleteEnvVar( const VarName : string) : Integer
10734:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10735:  Function ExpandEnvVars( const Str : string) : string
10736:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10737:  Procedure GetAllEnvVarNames( const Names : TStrings);
10738:  Function GetAllEnvVarNames1 : TStringDynArray;
10739:  Function EnvBlockSize : Integer
10740:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10741:   SIRegister_TPJEnvVarsEnumerator(CL);
10742:   SIRegister_TPJEnvVars(CL);
10743:   FindClass('TOBJECT'),'EPJEnvVars
10744:   FindClass('TOBJECT'),'EPJEnvVars
10745:  //Procedure Register
10746: end;
10747:
10748: (*----------------------------------------------------------------------------*)
10749: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10750: begin
10751:  'cOneSecInMS','LongInt'( 1000);
10752:  //'cDefTimeSlice','LongInt'( 50);
10753:  //'cDefMaxExecTime',' cOneMinInMS);
10754:  'cAppErrorMask','LongInt'( 1 shl 29);
10755:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10756:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10757:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10758:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10759:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10760:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10761:  Function MakeSize( const ACX, ACY : LongInt) : TSize
```

```
10762:   SIRegister_TPJCustomConsoleApp(CL);
10763:   SIRegister_TPJConsoleApp(CL);
10764: end;
10765:
10766: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10767: begin
10768:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10769:   t_encoding', '( uuencode, base64, mime )
10770:  Function internet_date( date : TDateTime) : string
10771:  Function lookup_hostname( const hostname : string) : longint
10772:  Function my_hostname : string
10773:  Function my_ip_address : longint
10774:  Function ip2string( ip_address : longint) : string
10775:  Function resolve_hostname( ip : longint) : string
10776:  Function address_from( const s : string; count : integer) : string
10777:  Function encode_base64( data : TStream) : TStringList
10778:  Function decode_base64( source : TStringList) : TMemoryStream
10779:  Function posn( const s, t : string; count : integer) : integer
10780:  Function poscn( c : char; const s : string; n : integer) : integer
10781:  Function filename_of( const s : string) : string
10782:  //Function trim( const s : string) : string
10783:  //Procedure setlength( var s : string; l : byte)
10784:  Function TimeZoneBias : longint
10785:  Function eight2seven_quoteprint( const s : string) : string
10786:  Function eight2seven_german( const s : string) : string
10787:  Function seven2eight_quoteprint( const s : string) : string end;
10788:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10789:  Function socketerror : cint
10790:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10791:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10792:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10793:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10794:  Function fplisten( s : cint; backlog : cint) : cint
10795:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10796:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10797:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10798:  Function NetAddrToStr( Entry : in_addr) : String
10799:  Function HostAddrToStr( Entry : in_addr) : String
10800:  Function StrToHostAddr( IP : String) : in_addr
10801:  Function StrToNetAddr( IP : String) : in_addr
10802:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10803:   cint8', 'shortint
10804:   cuint8', 'byte
10805:   cchar', 'cint8
10806:   cschar', 'cint8
10807:   cuchar', 'cuint8
10808:   cint16', 'smallint
10809:   cuint16', 'word
10810:   cshort', 'cint16
10811:   csshort', 'cint16
10812:   cushort', 'cuint16
10813:   cint32', 'longint
10814:   cuint32', 'longword
10815:   cint', 'cint32
10816:   csint', 'cint32
10817:   cuint', 'cuint32
10818:   csigned', 'cint
10819:   cunsigned', 'cuint
10820:   cint64', 'int64
10821:   clonglong', 'cint64
10822:   cslonglong', 'cint64
10823:   cbool', 'longbool
10824:   cfloat', 'single
10825:   cdouble', 'double
10826:   clongdouble', 'extended
10827:
10828: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10829: begin
10830:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10831:   SIRegister_TlkJSONdotnetclass(CL);
10832:   SIRegister_TlkJSONbase(CL);
10833:   SIRegister_TlkJSONnumber(CL);
10834:   SIRegister_TlkJSONstring(CL);
10835:   SIRegister_TlkJSONboolean(CL);
10836:   SIRegister_TlkJSONnull(CL);
10837:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10838:    +'se; data : TObject; var Continue : Boolean)
10839:   SIRegister_TlkJSONcustomlist(CL);
10840:   SIRegister_TlkJSONlist(CL);
10841:   SIRegister_TlkJSONobjectmethod(CL);
10842:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10843:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10844:   SIRegister_TlkHashTable(CL);
10845:   SIRegister_TlkBalTree(CL);
10846:   SIRegister_TlkJSONobject(CL);
10847:   SIRegister_TlkJSON(CL);
10848:   SIRegister_TlkJSONstreamed(CL);
10849:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10850: end;
```

```
10851:
10852: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10853: begin
10854:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10855:   SIRegister_TZSortedList(CL);
10856:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10857:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10858:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10859:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10860:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10861:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10862:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10863:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10864:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10865:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10866:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10867:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10868:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10869:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10870:  Function StrToBoolEx( Str : string) : Boolean
10871:  Function BoolToStrEx( Bool : Boolean) : String
10872:  Function IsIpAddr( const Str : string) : Boolean
10873:  Function zSplitString( const Str, Delimiters : string) : TStrings
10874:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10875:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10876:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10877:  Function FloatToSQLStr( Value : Extended) : string
10878:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10879:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10880:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10881:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10882:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10883:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10884:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10885:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10886:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10887:  Function BytesToVar( const Value : TByteDynArray) : Variant
10888:  Function VarToBytes( const Value : Variant) : TByteDynArray
10889:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10890:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10891:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10892:  Function EncodeCString( const Value : string) : string
10893:  Function DecodeCString( const Value : string) : string
10894:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10895:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10896:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10897:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10898:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10899:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10900:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10901:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10902:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10903:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10904:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10905:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10906:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10907:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10908: end;
10909:
10910: unit uPSI_ZEncoding;
10911:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10912:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10913:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10914:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10915:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10916:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10917:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10918:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10919:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10920:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10921:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10922:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10923:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10924:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10925:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10926:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10927:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10928:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10929:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10930:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10931:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10932:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10933:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10934:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10935:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10936:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10937:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
```

```
10938:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10939:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10940:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10941:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10942:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10943:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10944:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10945:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10946:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10947:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10948:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10949:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10950:  Function ZDefaultSystemCodePage : Word
10951:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10952:
10953:
10954: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10955: begin
10956:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10957:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10958:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10959:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10960:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10961:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10962:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10963:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10964:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10965:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10966:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10967:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10968:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10969:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10970:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10971:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10972:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10973:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10974:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10975:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10976:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10977:  ('ilIdentity','2 RPC_C_IMP_LEVEL_IDENTIFY);
10978:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10979:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10980:  ('EOAC_NONE','LongWord').SetUInt( $0);
10981:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10982:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10983:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10984:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10985:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10986:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10987:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10988:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10989:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10990:   FindClass('TOBJECT'),'EBoldCom
10991:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10992:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10993:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10994:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10995:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10996:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10997:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10998:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10999:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11000:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
11001:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
11002:  Function BoldCreateGUID : TGUID
11003:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
11004:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
11005:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11006:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11007: end;
11008:
11009: (*----------------------------------------------------------------------------*)
11010: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11011: begin
11012:  Function ParseISODate( s : string) : TDateTime
11013:  Function ParseISODateTime( s : string) : TDateTime
11014:  Function ParseISOTime( str : string) : TDateTime
11015: end;
11016:
11017: (*----------------------------------------------------------------------------*)
11018: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11019: begin
11020:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
11021:  Function BoldCreateGUIDWithBracketsAsString : string
11022: end;
11023:
11024: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11025: begin
```

```
11026:   FindClass('TOBJECT'),'TBoldFileHandler
11027:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11028:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11029:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
11030:   SIRegister_TBoldFileHandler(CL);
11031:   SIRegister_TBoldDiskFileHandler(CL);
11032:  Procedure BoldCloseAllFilehandlers
11033:  Procedure BoldRemoveUnchangedFilesFromEditor
11034:  Function BoldFileHandlerList : TBoldObjectArray
11035:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11036: end;
11037:
11038: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11039: begin
11040:   PCharArr', 'array of PChar
11041:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
11042:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11043:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11044:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11045:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
11046:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
11047:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
11048:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11049:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11050:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11051:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11052:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11053: end;
11054:
11055: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11056: begin
11057:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11058:   SIRegister_TfrmBoldQueryUser(CL);
11059:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11060: end;
11061:
11062: (*----------------------------------------------------------------------------*)
11063: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11064: begin
11065:  //('befIsInDisplayList',' BoldElementFlag0);
11066:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11067:  //('befFollowerSelected',' BoldElementFlag2);
11068:   FindClass('TOBJECT'),'TBoldQueue
11069:   FindClass('TOBJECT'),'TBoldQueueable
11070:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11071:   SIRegister_TBoldQueueable(CL);
11072:   SIRegister_TBoldQueue(CL);
11073:  Function BoldQueueFinalized : Boolean
11074:  Function BoldInstalledQueue : TBoldQueue
11075: end;
11076:
11077: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11078: begin
11079:  const mmPerInch','Extended').setExtended( 25.4);
11080:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11081:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11082:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11083:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11084:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11085:   TBarLineType', '( white, black, black_half )
11086:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11087:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11088:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11089:   TCheckSumMethod', '( csmNone, csmModulo10 )
11090:   SIRegister_TAsBarcode(CL);
11091:  Function CheckSumModulo10( const data : string) : string
11092:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11093:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11094:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11095:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11096: end;
11097:
11098: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11099: begin
11100:    THomogeneousByteVector', 'array[0..3] of Byte
11101:   THomogeneousWordVector', 'array[0..3] of Word
11102:   THomogeneousIntVector', 'array[0..3] of Integer
11103:   THomogeneousFltVector', 'array[0..3] of single
11104:   THomogeneousDblVector', 'array[0..3] of double
11105:   THomogeneousExtVector', 'array[0..3] of extended
11106:   TAffineByteVector', 'array[0..2] of Byte
11107:   TAffineWordVector', 'array[0..2] of Word
```

```
11108:   TAffineIntVector', 'array[0..2] of Integer
11109:   TAffineFltVector', 'array[0..2] of single
11110:   TAffineDblVector', 'array[0..2] of double
11111:   TAffineExtVector', 'array[0..2] of extended
11112:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11113:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11114:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11115:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11116:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11117:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11118:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11119:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11120:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11121:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11122:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11123:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11124:   TMatrix4b', 'THomogeneousByteMatrix
11125:   TMatrix4w', 'THomogeneousWordMatrix
11126:   TMatrix4i', 'THomogeneousIntMatrix
11127:   TMatrix4f', 'THomogeneousFltMatrix
11128:   TMatrix4d', 'THomogeneousDblMatrix
11129:   TMatrix4e', 'THomogeneousExtMatrix
11130:   TMatrix3b', 'TAffineByteMatrix
11131:   TMatrix3w', 'TAffineWordMatrix
11132:   TMatrix3i', 'TAffineIntMatrix
11133:   TMatrix3f', 'TAffineFltMatrix
11134:   TMatrix3d', 'TAffineDblMatrix
11135:   TMatrix3e', 'TAffineExtMatrix
11136:   //'PMatrix', '^TMatrix // will not work
11137:   TMatrixGL', 'THomogeneousFltMatrix
11138:   THomogeneousMatrix', 'THomogeneousFltMatrix
11139:   TAffineMatrix', 'TAffineFltMatrix
11140:   TQuaternion', 'record Vector : TVector4f; end
11141:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11142:   +'ger; Height : Integer; end
11143:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11144:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11145:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11146:  'EPSILON','Extended').setExtended( 1E-100);
11147:  'EPSILON2','Extended').setExtended( 1E-50);
11148:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11149:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11150:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11151:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11152:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11153:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11154:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11155:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11156:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11157:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11158:  Function VectorLength( V : array of Single) : Single
11159:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11160:  Procedure VectorNegate( V : array of Single)
11161:  Function VectorNorm( V : array of Single) : Single
11162:  Function VectorNormalize( V : array of Single) : Single
11163:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11164:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11165:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11166:  Procedure VectorScale( V : array of Single; Factor : Single)
11167:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11168:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11169:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11170:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11171:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11172:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11173:  Procedure MatrixAdjoint( var M : TMatrixGL)
11174:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11175:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11176:  Function MatrixDeterminant( M : TMatrixGL) : Single
11177:  Procedure MatrixInvert( var M : TMatrixGL)
11178:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11179:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11180:  Procedure MatrixTranspose( var M : TMatrixGL)
11181:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11182:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11183:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11184:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11185:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11186:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11187:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11188:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11189:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11190:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11191:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11192:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11193:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11194:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11195:  Function MakeAffineVector( V : array of Single) : TAffineVector
11196:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
```

```
11197:  Function MakeVector( V : array of Single) : TVectorGL
11198:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11199:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11200:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11201:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11202:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11203:  Function ArcCosGL( X : Extended) : Extended
11204:  Function ArcSinGL( X : Extended) : Extended
11205:  Function ArcTan2GL( Y, X : Extended) : Extended
11206:  Function CoTanGL( X : Extended) : Extended
11207:  Function DegToRadGL( Degrees : Extended) : Extended
11208:  Function RadToDegGL( Radians : Extended) : Extended
11209:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11210:  Function TanGL( X : Extended) : Extended
11211:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11212:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11213:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11214:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11215:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11216:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11217: end;
11218:
11219:
11220: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11221: begin
11222:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11223:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11224:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11225:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11226:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11227:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11228:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11229:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11230:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11231:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11232:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11233:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11234:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11235:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11236:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11237:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11238:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11239:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11240:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11241:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11242:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11243:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11244:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11245:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11246:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11247:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11248:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11249: end;
11250:
11251: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11252: begin
11253:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11254:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11255:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11256:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11257:   FindClass('TOBJECT'),'EInvalidParam
11258:  Function IsDCOMInstalled : Boolean
11259:  Function IsDCOMEnabled : Boolean
11260:  Function GetDCOMVersion : string
11261:  Function GetMDACVersion : string
11262:  Function GetMDACVersion2 : string
11263:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11264:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11265:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11266:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11267:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11268:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11269:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11270:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11271:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11272:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11273:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11274:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11275:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11276:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11277: end;
11278:
11279:
11280: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
```

```
11281: begin
11282:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11283:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11284:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11285:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11286:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11287:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11288:  DegPerCycle','Extended').setExtended( 360.0);
11289:  DegPerGrad','Extended').setExtended( 0.9);
11290:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11291:  GradPerCycle','Extended').setExtended( 400.0);
11292:  GradPerDeg','Extended').setExtended( 1.1111111111111111111111111111111);
11293:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11294:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11295:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11296:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11297:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11298:  CyclePerGrad','Extended').setExtended( 0.0025);
11299:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11300:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11301:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11302:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11303:  Function MakePercentage( const Step, Max : Longint) : Longint
11304:  Function CelsiusToKelvin( const T : double) : double
11305:  Function CelsiusToFahrenheit( const T : double) : double
11306:  Function KelvinToCelsius( const T : double) : double
11307:  Function KelvinToFahrenheit( const T : double) : double
11308:  Function FahrenheitToCelsius( const T : double) : double
11309:  Function FahrenheitToKelvin( const T : double) : double
11310:  Function CycleToDeg( const Cycles : double) : double
11311:  Function CycleToGrad( const Cycles : double) : double
11312:  Function CycleToRad( const Cycles : double) : double
11313:  Function DegToCycle( const Degrees : double) : double
11314:  Function DegToGrad( const Degrees : double) : double
11315:  Function DegToRad( const Degrees : double) : double
11316:  Function GradToCycle( const Grads : double) : double
11317:  Function GradToDeg( const Grads : double) : double
11318:  Function GradToRad( const Grads : double) : double
11319:  Function RadToCycle( const Radians : double) : double
11320:  Function RadToDeg( const Radians : double) : double
11321:  Function RadToGrad( const Radians : double) : double
11322:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11323:  Function DmsToRad( const D, M : Integer; const S : double) : double
11324:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11325:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11326:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11327:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11328:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11329:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11330:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11331:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11332:  Function CmToInch( const Cm : double) : double
11333:  Function InchToCm( const Inch : double) : double
11334:  Function FeetToMetre( const Feet : double) : double
11335:  Function MetreToFeet( const Metre : double) : double
11336:  Function YardToMetre( const Yard : double) : double
11337:  Function MetreToYard( const Metre : double) : double
11338:  Function NmToKm( const Nm : double) : double
11339:  Function KmToNm( const Km : double) : double
11340:  Function KmToSm( const Km : double) : double
11341:  Function SmToKm( const Sm : double) : double
11342:  Function LitreToGalUs( const Litre : double) : double
11343:  Function GalUsToLitre( const GalUs : double) : double
11344:  Function GalUsToGalCan( const GalUs : double) : double
11345:  Function GalCanToGalUs( const GalCan : double) : double
11346:  Function GalUsToGalUk( const GalUs : double) : double
11347:  Function GalUkToGalUs( const GalUk : double) : double
11348:  Function LitreToGalCan( const Litre : double) : double
11349:  Function GalCanToLitre( const GalCan : double) : double
11350:  Function LitreToGalUk( const Litre : double) : double
11351:  Function GalUkToLitre( const GalUk : double) : double
11352:  Function KgToLb( const Kg : double) : double
11353:  Function LbToKg( const Lb : double) : double
11354:  Function KgToOz( const Kg : double) : double
11355:  Function OzToKg( const Oz : double) : double
11356:  Function CwtUsToKg( const Cwt : double) : double
11357:  Function CwtUkToKg( const Cwt : double) : double
11358:  Function KaratToKg( const Karat : double) : double
11359:  Function KgToCwtUs( const Kg : double) : double
11360:  Function KgToCwtUk( const Kg : double) : double
11361:  Function KgToKarat( const Kg : double) : double
11362:  Function KgToSton( const Kg : double) : double
11363:  Function KgToLton( const Kg : double) : double
11364:  Function StonToKg( const STon : double) : double
11365:  Function LtonToKg( const Lton : double) : double
11366:  Function QrUsToKg( const Qr : double) : double
11367:  Function QrUkToKg( const Qr : double) : double
11368:  Function KgToQrUs( const Kg : double) : double
11369:  Function KgToQrUk( const Kg : double) : double
```

```
11370:  Function PascalToBar( const Pa : double) : double
11371:  Function PascalToAt( const Pa : double) : double
11372:  Function PascalToTorr( const Pa : double) : double
11373:  Function BarToPascal( const Bar : double) : double
11374:  Function AtToPascal( const At : double) : double
11375:  Function TorrToPascal( const Torr : double) : double
11376:  Function KnotToMs( const Knot : double) : double
11377:  Function HpElectricToWatt( const HpE : double) : double
11378:  Function HpMetricToWatt( const HpM : double) : double
11379:  Function MsToKnot( const ms : double) : double
11380:  Function WattToHpElectric( const W : double) : double
11381:  Function WattToHpMetric( const W : double) : double
11382:  function getBigPI: string;    //PI of 1000 numbers
11383:
11384: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11385: begin
11386:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11387:  Procedure CDCopyFile( const FileName, DestName : string)
11388:  Procedure CDMoveFile( const FileName, DestName : string)
11389:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11390:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11391:  Function CDGetTempDir : string
11392:  Function CDGetFileSize( FileName : string) : longint
11393:  Function GetFileTime( FileName : string) : longint
11394:  Function GetShortName( FileName : string) : string
11395:  Function GetFullName( FileName : string) : string
11396:  Function WinReboot : boolean
11397:  Function WinDir : String
11398:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11399:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11400:  Function devExecutor : TdevExecutor
11401: end;
11402:
11403: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11404: begin
11405:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11406:  Procedure Associate( Index : integer)
11407:  Procedure UnAssociate( Index : integer)
11408:  Function IsAssociated( Index : integer) : boolean
11409:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11410:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11411:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11412:  procedure RefreshIcons;
11413: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11414: function MergColor(Colors: Array of TColor): TColor;
11415: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11416: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11417: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11418: function GetInverseColor(AColor: TColor): TColor;
11419: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11420: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11421: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11422: Procedure GetSystemMenuFont(Font: TFont);
11423: end;
11424:
11425: //*************************unit uPSI_JvHLParser;*******************************
11426: function IsStringConstant(const St: string): Boolean;
11427: function IsIntConstant(const St: string): Boolean;
11428: function IsRealConstant(const St: string): Boolean;
11429: function IsIdentifier(const ID: string): Boolean;
11430: function GetStringValue(const St: string): string;
11431: procedure ParseString(const S: string; Ss: TStrings);
11432: function IsStringConstantW(const St: WideString): Boolean;
11433: function IsIntConstantW(const St: WideString): Boolean;
11434: function IsRealConstantW(const St: WideString): Boolean;
11435: function IsIdentifierW(const ID: WideString): Boolean;
11436: function GetStringValueW(const St: WideString): WideString;
11437: procedure ParseStringW(const S: WideString; Ss: TStrings);
11438:
11439:
11440: //*************************unit uPSI_JclMapi;*******************************
11441:
11442: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11443: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11444: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11445: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11446: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11447:
11448: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11449: begin
11450:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11451:  Function BuildType1Message( ADomain, AHost : String) : String
11452:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11453:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11454:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11455: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

```
11456: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11457: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11458: end;
11459:
11460: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11461: begin
11462: ('IpAny','LongWord').SetUInt( $00000000);
11463:  IpLoopBack','LongWord').SetUInt( $7F000001);
11464:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11465:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11466:  PortAny','LongWord( $0000);
11467:  SocketMaxConnections','LongInt'( 5);
11468:  TIpAddr', 'LongWord
11469:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11470:  Function HostToNetLong( HostLong : LongWord) : LongWord
11471:  Function HostToNetShort( HostShort : Word) : Word
11472:  Function NetToHostLong( NetLong : LongWord) : LongWord
11473:  Function NetToHostShort( NetShort : Word) : Word
11474:  Function StrToIp( Ip : string) : TIpAddr
11475:  Function IpToStr( Ip : TIpAddr) : string
11476: end;
11477:
11478: (*---------------------------------------------------------------------------*)
11479: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11480: begin
11481:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11482:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11483:    +'amSha1, AlsmtpClientAuthAutoSelect )
11484:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11485:   SIRegister_TAlSmtpClient(CL);
11486: end;
11487:
11488: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11489: begin
11490: 'TBitNo', 'Integer
11491:   TStByteNo', 'Integer
11492: TStationNo', 'Integer
11493: TInOutNo', 'Integer
11494: TIo', '( EE, AA, NE, NA )
11495: TBitSet', 'set of TBitNo
11496: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11497: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11498: TBitAddr', 'LongInt
11499: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11500: TByteAddr', 'SmallInt
11501: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11502:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11503:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11504:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
      aBitNo:TBitNo);
11505:  Function BitAddrToStr( Value : TBitAddr) : string
11506:  Function StrToBitAddr( const Value : string) : TBitAddr
11507:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11508:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11509:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11510:  Function ByteAddrToStr( Value : TByteAddr) : string
11511:  Function StrToByteAddr( const Value : string) : TByteAddr
11512:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11513:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11514:  Function InOutStateToStr( State : TInOutState) : string
11515:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11516:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11517: end;
11518:
11519: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11520: begin
11521: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11522:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11523: DpmiPmVector', 'Int64
11524:  'DInterval','LongInt'( 1000);
11525:  //'DEnabled','Boolean')BoolToStr( True);
11526:  'DIntFreq','string' if64
11527:  //'DMessages','Boolean if64);
11528:   SIRegister_TwdxCustomTimer(CL);
11529:   SIRegister_TwdxTimer(CL);
11530:   SIRegister_TwdxRtcTimer(CL);
11531:   SIRegister_TCustomIntTimer(CL);
11532:   SIRegister_TIntTimer(CL);
11533:   SIRegister_TRtcIntTimer(CL);
11534:  Function RealNow : TDateTime
11535:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11536:  Function DateTimeToMs( Time : TDateTime) : LongInt
11537: end;
11538:
11539: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11540: begin
11541: TIdSyslogPRI', 'Integer
11542: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11543:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
```

```
11544:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11545:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11546:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11547: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11548:   SIRegister_TIdSysLogMsgPart(CL);
11549:   SIRegister_TIdSysLogMessage(CL);
11550:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11551:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11552:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11553:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11554:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11555:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11556: end;
11557:
11558: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11559: begin
11560:  'UWhitespace','String '(?:\s*)
11561:  Function StripSpaces( const AText : string) : string
11562:  Function CharCount( const AText : string; Ch : Char) : Integer
11563:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11564:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11565: end;
11566:
11567:
11568: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11569: begin
11570:  ExtPascalVersion','String '0.9.8
11571:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11572:    +'Opera, brKonqueror, brMobileSafari )
11573:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11574:   AddTypeS('TExtProcedure', 'Procedure
11575:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11576:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11577:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11578:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11579:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11580:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11581:  Function StrToJS( const S : string; UseBR : boolean) : string
11582:  Function CaseOf( const S : string; const Cases : array of string) : integer
11583:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11584:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11585:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11586:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11587:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11588:  Function IsUpperCase( S : string) : boolean
11589:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11590:  Function BeautifyCSS( const AStyle : string) : string
11591:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11592:  Function JSDateToDateTime( JSDate : string) : TDateTime
11593: end;
11594:
11595: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11596: begin
11597:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11598:   TSHDeleteOptions', 'set of TSHDeleteOption
11599:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11600:   TSHRenameOptions', 'set of TSHRenameOption
11601:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11602:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11603:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11604:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11605:   TEnumFolderFlags', 'set of TEnumFolderFlag
11606:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11607:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11608:    +'IEnumIdList; Folder : IShellFolder; end
11609:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11610:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11611:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11612:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11613:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11614:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11615:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11616:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11617:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11618:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11619:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11620:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11621:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11622:  Function SHFreeMem( var P : Pointer) : Boolean
11623:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11624:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11625:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11626:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11627:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11628:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11629:  Function PidlFree( var IdList : PItemIdList) : Boolean
11630:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11631:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11632:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
```

```
11633:  Function PidlToPath( IdList : PItemIdList) : string
11634:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11635:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11636:   PShellLink', '^TShellLink // will not work
11637:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11638:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11639:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11640:  Procedure ShellLinkFree( var Link : TShellLink)
11641:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11642:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11643:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11644:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11645:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11646:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11647:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11648:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11649:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11650:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11651:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11652:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11653:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11654:  Function ShellOpenAs( const FileName : string) : Boolean
11655:  Function ShellRasDial( const EntryName : string) : Boolean
11656:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11657:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11658:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11659:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11660:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11661:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11662:  Function OemKeyScan( wOemChar : Word) : DWORD
11663:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11664: end;
11665:
11666: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11667: begin
11668:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11669:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11670:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11671:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11672:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11673:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11674:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11675:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11676:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11677:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11678:  Function xmlValidName( const Text : UnicodeString) : Boolean
11679:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11680:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11681:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11682:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11683:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
     : TUnicodeCodecClass
11684:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11685:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11686:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11687:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11688:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11689:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11690:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11691:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11692:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11693:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11694:  Procedure SelfTestcXMLFunctions
11695: end;
11696:
11697: (*----------------------------------------------------------------------------*)
11698: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11699: begin
11700:  Function AWaitCursor : IUnknown
11701:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11702:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11703:  Function YesNo( const ACaption, AMsg : string) : boolean
11704:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11705:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11706:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11707:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11708:  Procedure GetSystemPaths( Strings : TStrings)
11709:  Procedure MakeEditNumeric( EditHandle : integer)
11710: end;
11711:
11712: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11713: begin
11714:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11715:  'BI_YUY2','LongWord( $32595559);
11716:  'BI_UYVY','LongWord').SetUInt( $59565955);
11717:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11718:  'BI_YVU9','LongWord').SetUInt( $39555659);
11719:  'BI_YUV12','LongWord( $30323449);
11720:  'BI_Y8','LongWord').SetUInt( $20203859);
```

```
11721:  'BI_Y211','LongWord').SetUInt( $31313259);
11722:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11723:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11724: end;
11725:
11726: (*----------------------------------------------------------------------------*)
11727: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11728: begin
11729:  'WM_USER','LongWord').SetUInt( $0400);
11730:  'WM_CAP_START','LongWord').SetUint($0400);
11731:  'WM_CAP_END','longword').SetUint($0400+85);
11732:  //WM_CAP_START+  85
11733:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11734:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11735:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11736:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11737:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11738:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11739:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11740:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11741:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11742:  Function capGetUserData( hwnd : THandle) : LongInt
11743:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11744:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11745:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11746:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11747:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11748:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11749:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11750:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11751:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11752:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11753:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11754:  Function capEditCopy( hwnd : THandle) : LongInt
11755:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11756:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11757:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11758:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11759:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11760:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11761:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11762:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11763:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11764:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11765:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11766:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11767:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11768:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11769:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11770:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11771:  Function capGrabFrame( hwnd : THandle) : LongInt
11772:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11773:  Function capCaptureSequence( hwnd : THandle) : LongInt
11774:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11775:  Function capCaptureStop( hwnd : THandle) : LongInt
11776:  Function capCaptureAbort( hwnd : THandle) : LongInt
11777:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11778:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11779:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11780:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11781:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11782:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11783:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11784:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11785:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11786:  Function capPalettePaste( hwnd : THandle) : LongInt
11787:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11788:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11789:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11790:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11791:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11792:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11793:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11794:   //PCapStatus', '^TCapStatus // will not work
11795:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11796:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11797:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11798:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11799:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11800:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11801:    +' wNumAudioAllocated : WORD; end
11802:   //PCaptureParms', '^TCaptureParms // will not work
11803:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11804:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11805:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11806:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11807:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11808:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11809:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
```

```
11810:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11811:    +'he : BOOL; AVStreamMaster : WORD; end
11812:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11813:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11814:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11815:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11816:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
       : Integer; hwndParent : THandle; nID : Integer) : THandle
11817:  Function
       capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11818:  'IDS_CAP_BEGIN','LongInt'( 300);
11819:  'IDS_CAP_END','LongInt'( 301);
11820:  'IDS_CAP_INFO','LongInt'( 401);
11821:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11822:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11823:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11824:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11825:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11826:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11827:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11828:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11829:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11830:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11831:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11832:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11833:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11834:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11835:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11836:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11837:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11838:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11839:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11840:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11841:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11842:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11843:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11844:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11845:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11846:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11847:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11848:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11849:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11850:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11851:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11852:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11853:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11854:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11855:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11856:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11857:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11858:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11859:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11860:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11861:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11862:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11863:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11864:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11865:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11866:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11867:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11868:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11869:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11870:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11871:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11872:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11873:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11874:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11875:  'AVICAP32','String 'AVICAP32.dll
11876: end;
11877:
11878: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11879: begin
11880:  Function AlBoolToInt( Value : Boolean) : Integer
11881:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11882:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11883:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11884:  Function ALInc( var x : integer; Count : integer) : Integer
11885:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11886:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11887:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11888:  Function ALIsInteger(const S: AnsiString): Boolean;
11889:  function ALIsDecimal(const S: AnsiString): boolean;
11890:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11891:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11892:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11893:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11894:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11895:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11896:  Function ALRandomStr(const aLength: Longint): AnsiString;
```

```
11897:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11898:  Function ALRandomStrU(const aLength: Longint): String;
11899: end;
11900:
11901: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11902: begin
11903:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11904: end;
11905:
11906: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11907: begin
11908:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11909:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11910:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11911:    +'; ullAvailExtendedVirtual : Int64; end
11912:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11913:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11914:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11915:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11916:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11917:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11918:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11919:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11920: end;
11921:
11922: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11923: begin
11924:   SIRegister_THandledObject(CL);
11925:   SIRegister_TEvent(CL);
11926:   SIRegister_TMutex(CL);
11927:   SIRegister_TSharedMem(CL);
11928:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11929:  'TRACE_BUFFER','String 'TRACE_BUFFER
11930:  'TRACE_MUTEX','String 'TRACE_MUTEX
11931:   //PTraceEntry', '^TTraceEntry // will not work
11932:   SIRegister_TIPCTracer(CL);
11933:  'MAX_CLIENTS','LongInt'( 6);
11934:  'IPCTIMEOUT','LongInt'( 2000);
11935:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11936:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11937:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11938:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11939:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11940:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11941:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11942:   FindClass('TOBJECT'),'EMonitorActive
11943:   FindClass('TOBJECT'),'TIPCThread
11944:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11945:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11946:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11947:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11948:   TClientFlags', 'set of TClientFlag
11949:   //PEventData', '^TEventData // will not work
11950:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11951:    +'lag; Flags : TClientFlags; end
11952:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11953:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11954:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11955:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11956:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11957:   SIRegister_TIPCEvent(CL);
11958:   //PClientDirRecords', '^TClientDirRecords // will not work
11959:   SIRegister_TClientDirectory(CL);
11960:   TIPCState', '( stInActive, stDisconnected, stConnected )
11961:   SIRegister_TIPCThread(CL);
11962:   SIRegister_TIPCMonitor(CL);
11963:   SIRegister_TIPCClient(CL);
11964:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11965: end;
11966:
11967: (*----------------------------------------------------------------------------*)
11968: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11969: begin
11970:   SIRegister_TAlGSMComm(CL);
11971:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11972:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
       AMessage:AnsiString);
11973:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11974:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
       UseGreekAlphabet:Bool):Widestring;
11975:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11976:  end;
11977:
11978: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11979: begin
11980:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11981:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11982:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
```

```
11983:   TInternetScheme', 'integer
11984:   TALIPv6Binary', 'array[1..16] of Char;
11985:  // TALIPv6Binary = array[1..16] of ansiChar;
11986:  //   TInternetScheme = Integer;
11987:   SIRegister_TALHTTPRequestHeader(CL);
11988:   SIRegister_TALHTTPCookie(CL);
11989:   SIRegister_TALHTTPCookieCollection(CL);
11990:   SIRegister_TALHTTPResponseHeader(CL);
11991:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11992:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11993: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
       Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11994: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
       Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11995: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11996:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11997:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11998:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11999:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
12000:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
12001:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
       ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
12002:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
       Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
12003:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
12004:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
12005:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
12006:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
12007:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
12008:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
12009:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
12010:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
12011:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
12012:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
12013:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
12014:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
12015:  Function ALZeroIpV6 : TALIPv6Binary
12016:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
12017:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
12018:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
12019:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
12020: end;
12021:
12022: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
12023: begin
12024:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
12025:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12026:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
12027:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12028:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
12029:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
12030:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
12031:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
12032:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
12033:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12034:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
12035: end;
12036:
12037: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12038: begin
12039:   SIRegister_TALEMailHeader(CL);
12040:   SIRegister_TALNewsArticleHeader(CL);
12041:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
12042:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
12043:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
12044:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
12045:  Function AlGenerateInternetMessageID : AnsiString;
12046:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
12047:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
12048:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12049: end;
12050:
12051: (*----------------------------------------------------------------------------*)
12052: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12053: begin
12054:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12055:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
12056:  Function ALgetLocalIPs : TALStrings
12057:  Function ALgetLocalHostName : AnsiString
12058: end;
12059:
12060: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12061: begin
12062:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
```

```
12063:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12064:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12065:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12066:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12067:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12068:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12069:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12070:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
12071: end;
12072:
12073: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12074: begin
12075:   TStartupInfoA', 'TStartupInfo
12076:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12077:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12078:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12079:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12080:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12081:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12082:  SE_TCB_NAME','String 'SeTcbPrivilege
12083:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12084:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12085:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12086:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12087:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12088:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12089:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12090:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12091:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12092:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12093:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12094:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12095:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12096:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12097:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12098:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12099:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12100:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12101:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12102:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12103:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12104:  Function AlGetEnvironmentString : AnsiString
12105:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12106:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12107:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12108:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12109:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12110: end;
12111:
12112: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12113: begin
12114:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12115:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12116:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12117:  Function ALGetModuleName : ansistring
12118:  Function ALGetModuleFileNameWithoutExtension : ansistring
12119:  Function ALGetModulePath : ansiString
12120:  Function AlGetFileSize( const AFileName : ansistring) : int64
12121:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12122:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12123:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12124:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12125:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12126:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12127:  Function ALFileExists( const Path : ansiString) : boolean
12128:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12129:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12130:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12131:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12132:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12133: end;
12134:
12135: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12136: begin
12137:   NativeInt', 'Integer
12138:   NativeUInt', 'Cardinal
```

```
12139:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12140:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12141:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12142:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12143:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12144:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12145:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12146:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12147:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12148:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12149:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12150:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12151:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12152:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12153:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12154:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12155:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12156:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12157:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12158:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12159:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12160:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12161:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12162:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12163:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12164:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12165:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12166:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12167:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12168:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12169:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12170:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12171: end;
12172:
12173: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12174: begin
12175:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12176:   FindClass('TOBJECT'),'TALXMLNode
12177:   FindClass('TOBJECT'),'TALXMLNodeList
12178:   FindClass('TOBJECT'),'TALXMLDocument
12179:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12180:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12181:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12182:    +'nst Name : AnsiString; const Attributes : TALStrings)
12183:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12184:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12185:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12186:    +'ntDocType, ntDocFragment, ntNotation )
12187:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12188:   TALXMLDocOptions', 'set of TALXMLDocOption
12189:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12190:   TALXMLParseOptions', 'set of TALXMLParseOption
12191:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12192:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12193:   SIRegister_EALXMLDocError(CL);
12194:   SIRegister_TALXMLNodeList(CL);
12195:   SIRegister_TALXMLNode(CL);
12196:   SIRegister_TALXmlElementNode(CL);
12197:   SIRegister_TALXmlAttributeNode(CL);
12198:   SIRegister_TALXmlTextNode(CL);
12199:   SIRegister_TALXmlDocumentNode(CL);
12200:   SIRegister_TALXmlCommentNode(CL);
12201:   SIRegister_TALXmlProcessingInstrNode(CL);
12202:   SIRegister_TALXmlCDataNode(CL);
12203:   SIRegister_TALXmlEntityRefNode(CL);
12204:   SIRegister_TALXmlEntityNode(CL);
12205:   SIRegister_TALXmlDocTypeNode(CL);
12206:   SIRegister_TALXmlDocFragmentNode(CL);
12207:   SIRegister_TALXmlNotationNode(CL);
12208:   SIRegister_TALXMLDocument(CL);
12209:  cAlXMLUTF8EncodingStr','String 'UTF-8
12210:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12211:  CALNSDelim','String ':
12212:  CALXML','String 'xml
12213:  CALVersion','String 'version
12214:  CALEncoding','String 'encoding
12215:  CALStandalone','String 'standalone
```

```
12216:  CALDefaultNodeIndent','String '
12217:  CALXmlDocument','String 'DOCUMENT
12218:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12219:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
        EncodingStr:AnsiString);
12220:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
        ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12221:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
        ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12222:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
        Recurse: Boolean):TalxmlNode
12223:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
        AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12224:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
        AnsiString
12225: end;
12226:
12227: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12228: //based on TEEProc, TeCanvas, TEEngine, TChart
12229: begin
12230:  'TeePiStep','Double').setExtended( Pi / 180.0);
12231:  'TeeDefaultPerspective','LongInt'( 100);
12232:  'TeeMinAngle','LongInt'( 270);
12233:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12234:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12235:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12236:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12237:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12238:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12239:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12240:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12241:  'TA_LEFT','LongInt'( 0);
12242:  'TA_RIGHT','LongInt'( 2);
12243:  'TA_CENTER','LongInt'( 6);
12244:  'TA_TOP','LongInt'( 0);
12245:  'TA_BOTTOM','LongInt'( 8);
12246:  'teePATCOPY','LongInt'( 0);
12247:  'NumCirclePoints','LongInt'( 64);
12248:  'teeDEFAULT_CHARSET','LongInt'( 1);
12249:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12250:  'TA_LEFT','LongInt'( 0);
12251:  'bs_Solid','LongInt'( 0);
12252:  'teepf24Bit','LongInt'( 0);
12253:  'teepfDevice','LongInt'( 1);
12254:  'CM_MOUSELEAVE','LongInt'( 10000);
12255:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12256:  'DC_BRUSH','LongInt'( 18);
12257:  'DC_PEN','LongInt'( 19);
12258:   teeCOLORREF', 'LongWord
12259:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12260:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12261:   SIRegister_TFilterRegion(CL);
12262:   SIRegister_IFormCreator(CL);
12263:   SIRegister_TTeeFilter(CL);
12264:   //TFilterClass', 'class of TTeeFilter
12265:   SIRegister_TFilterItems(CL);
12266:   SIRegister_TConvolveFilter(CL);
12267:   SIRegister_TBlurFilter(CL);
12268:   SIRegister_TTeePicture(CL);
12269:   TPenEndStyle', '( esRound, esSquare, esFlat )
12270:   SIRegister_TChartPen(CL);
12271:   SIRegister_TChartHiddenPen(CL);
12272:   SIRegister_TDottedGrayPen(CL);
12273:   SIRegister_TDarkGrayPen(CL);
12274:   SIRegister_TWhitePen(CL);
12275:   SIRegister_TChartBrush(CL);
12276:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12277:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12278:   SIRegister_TView3DOptions(CL);
12279:   FindClass('TOBJECT'),'TTeeCanvas
12280:   TTeeTransparency', 'Integer
12281:   SIRegister_TTeeBlend(CL);
12282:   FindClass('TOBJECT'),'TCanvas3D
12283:   SIRegister_TTeeShadow(CL);
12284:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
        gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12285:   FindClass('TOBJECT'),'TSubGradient
12286:   SIRegister_TCustomTeeGradient(CL);
12287:   SIRegister_TSubGradient(CL);
12288:   SIRegister_TTeeGradient(CL);
12289:   SIRegister_TTeeFontGradient(CL);
12290:   SIRegister_TTeeFont(CL);
12291:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12292:   TCanvasTextAlign', 'Integer
12293:   TTeeCanvasHandle', 'HDC
12294:   SIRegister_TTeeCanvas(CL);
12295:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12296:   SIRegister_TFloatXYZ(CL);
12297:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
```

```
12298:   TRGB', 'record blue: byte; green: byte; red: byte; end
12299:   {TRGB=packed record
12300:     Blue  : Byte;
12301:     Green : Byte;
12302:     Red   : Byte;
12303:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12304:
12305: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12306:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12307: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12308: TCanvas3DPlane', '( cpX, cpY, cpZ )
12309:   //IInterface', 'IUnknown
12310:   SIRegister_TCanvas3D(CL);
12311:   SIRegister_TTeeCanvas3D(CL);
12312:   TTrianglePoints', 'Array[0..2] of TPoint;
12313:   TFourPoints', 'Array[0..3] of TPoint;
12314:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12315:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12316:  Function Point3D( const x, y, z : Integer) : TPoint3D
12317:  Procedure SwapDouble( var a, b : Double)
12318:  Procedure SwapInteger( var a, b : Integer)
12319:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12320:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12321:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12322:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12323:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12324:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12325:  Procedure UnClipCanvas( ACanvas : TCanvas)
12326:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12327:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12328:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12329:  'TeeCharForHeight','String 'W
12330:  'DarkerColorQuantity','Byte').SetUInt( 128);
12331:  'DarkColorQuantity','Byte').SetUInt( 64);
12332:   TButtonGetColorProc', 'Function  : TColor
12333:   SIRegister_TTeeButton(CL);
12334:   SIRegister_TButtonColor(CL);
12335:   SIRegister_TComboFlat(CL);
12336:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12337:  Function TeePoint( const aX, aY : Integer) : TPoint
12338:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12339:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12340:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12341:  Function OrientRectangle( const R : TRect) : TRect
12342:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12343:  Function PolygonBounds( const P : array of TPoint) : TRect
12344:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12345:  Function RGBValue( const Color : TColor) : TRGB
12346:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12347:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12348:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12349:  Function TeeCull( const P : TFourPoints) : Boolean;
12350:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12351:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12352:  Procedure SmoothStretch( Src, Dst : TBitmap);
12353:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12354:  Function TeeDistance( const x, y : Double) : Double
12355:  Function TeeLoadLibrary( const FileName : String) : HInst
12356:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12357:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12358:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12359:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
       Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12360:   SIRegister_ICanvasHyperlinks(CL);
12361:   SIRegister_ICanvasToolTips(CL);
12362:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12363: end;
12364:
12365: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12366: begin
12367:   TOvcHdc', 'Integer
12368:   TOvcHWND', 'Cardinal
12369:   TOvcHdc', 'HDC
12370:   TOvcHWND', 'HWND
12371:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12372:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12373:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12374:  Function DefaultEpoch : Integer
12375:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12376:  Procedure FixRealPrim( P : PChar; DC : Char)
12377:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12378:  Function GetLeftButton : Byte
12379:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12380:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12381:  Function GetShiftFlags : Byte
12382:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12383:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12384:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12385:  Function ovIsForegroundTask : Boolean
```

```
12386:  Function ovTrimLeft( const S : string) : string
12387:  Function ovTrimRight( const S : string) : string
12388:  Function ovQuotedStr( const S : string) : string
12389:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12390:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12391:  Function PtrDiff( const P1, P2 : PChar) : Word
12392:  Procedure PtrInc( var P, Delta : Word)
12393:  Procedure PtrDec( var P, Delta : Word)
12394:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12395:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
       SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12396:  Function ovMinI( X, Y : Integer) : Integer
12397:  Function ovMaxI( X, Y : Integer) : Integer
12398:  Function ovMinL( X, Y : LongInt) : LongInt
12399:  Function ovMaxL( X, Y : LongInt) : LongInt
12400:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12401:  Function PartialCompare( const S1, S2 : string) : Boolean
12402:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12403:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12404:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12405:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
       TransparentColor : TColor)
12406:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
       TransparentColor : TColorRef)
12407:  Function ovWidthOf( const R : TRect) : Integer
12408:  Function ovHeightOf( const R : TRect) : Integer
12409:  Procedure ovDebugOutput( const S : string)
12410:  Function GetArrowWidth( Width, Height : Integer) : Integer
12411:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12412:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12413:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12414:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12415:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12416:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12417:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12418:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12419:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12420:  Function CreateMetaFile( p1 : PChar) : HDC
12421:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12422:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12423:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12424:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12425:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12426:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12427:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12428:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12429:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12430:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12431:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12432:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12433:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12434:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12435:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12436:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12437:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12438:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12439:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12440:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12441:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12442:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12443:  Function UpdateColors( DC : HDC) : BOOL
12444:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12445:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12446:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12447:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12448:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12449:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12450:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
       HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12451:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12452:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12453:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12454:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12455:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12456:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12457:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12458:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12459:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12460:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12461:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12462:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12463:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12464: end;
12465:
12466: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12467: begin
```

```
12468:   SIRegister_TOvcAbstractStore(CL);
12469:   //PExPropInfo', '^TExPropInfo // will not work
12470: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12471:   SIRegister_TOvcPropertyList(CL);
12472:   SIRegister_TOvcDataFiler(CL);
12473:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12474:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12475:  Function CreateStoredItem( const CompName, PropName : string) : string
12476:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12477:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12478: end;
12479:
12480: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12481: begin
12482:  'ovsetsize','LongInt'( 16);
12483:  'etSyntax','LongInt'( 0);
12484:  'etSymantic','LongInt'( 1);
12485:  'chCR','Char #13);
12486:  'chLF','Char #10);
12487:  'chLineSeparator',' chCR);
12488:   SIRegister_TCocoError(CL);
12489:   SIRegister_TCommentItem(CL);
12490:   SIRegister_TCommentList(CL);
12491:   SIRegister_TSymbolPosition(CL);
12492: TGenListType', '( glNever, glAlways, glOnError )
12493: TovBitSet', 'set of Integer
12494:   //PStartTable', '^TStartTable // will not work
12495: 'TovCharSet', 'set of AnsiChar
12496: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12497: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12498: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12499: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12500: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12501:    +'osition; const Data : string; ErrorType : integer)
12502: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12503: TGetCH', 'Function ( pos : longint) : char
12504: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12505:   SIRegister_TCocoRScanner(CL);
12506:   SIRegister_TCocoRGrammar(CL);
12507:  '_EF','Char #0);
12508:  '_TAB','Char').SetString( #09);
12509:  '_CR','Char').SetString( #13);
12510:  '_LF','Char').SetString( #10);
12511:  '_EL','').SetString( _CR);
12512:  '_EOF','Char').SetString( #26);
12513:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12514:  'minErrDist','LongInt'( 2);
12515:  Function ovPadL( S : string; ch : char; L : integer) : string
12516: end;
12517:
12518:   TFormatSettings = record
12519:     CurrencyFormat: Byte;
12520:     NegCurrFormat: Byte;
12521:     ThousandSeparator: Char;
12522:     DecimalSeparator: Char;
12523:     CurrencyDecimals: Byte;
12524:     DateSeparator: Char;
12525:     TimeSeparator: Char;
12526:     ListSeparator: Char;
12527:     CurrencyString: string;
12528:     ShortDateFormat: string;
12529:     LongDateFormat: string;
12530:     TimeAMString: string;
12531:     TimePMString: string;
12532:     ShortTimeFormat: string;
12533:     LongTimeFormat: string;
12534:     ShortMonthNames: array[1..12] of string;
12535:     LongMonthNames: array[1..12] of string;
12536:     ShortDayNames: array[1..7] of string;
12537:     LongDayNames: array[1..7] of string;
12538:     TwoDigitYearCenturyWindow: Word;
12539:   end;
12540:
12541: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12542: begin
12543:  Function ovFormatSettings : TFormatSettings
12544: end;
12545:
12546: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12547: begin
12548:   TOvcCharSet', 'set of Char
12549:   ovBTable', 'array[0..255] of Byte
12550:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
      HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12551:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12552:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12553:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12554:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12555:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
```

```
12556:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12557:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12558:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12559:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12560:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12561:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12562:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12563:  Function LoCaseChar( C : Char) : Char
12564:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12565:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12566:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12567:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12568:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12569:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12570:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12571:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12572:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12573:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12574:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12575:  Procedure TrimAllSpacesPChar( P : PChar)
12576:  Function TrimEmbeddedZeros( const S : string) : string
12577:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12578:  Function TrimTrailPrimPChar( S : PChar) : PChar
12579:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12580:  Function TrimTrailingZeros( const S : string) : string
12581:  Procedure TrimTrailingZerosPChar( P : PChar)
12582:  Function UpCaseChar( C : Char) : Char
12583:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12584:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12585:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12586: end;
12587:
12588: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12589: begin
12590:   //PRaiseFrame', '^TRaiseFrame // will not work
12591:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12592:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12593:  Procedure SafeCloseHandle( var Handle : THandle)
12594:  Procedure ExchangeInteger( X1, X2 : Integer)
12595:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12596:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12597:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12598:
12599: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12600:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12601: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12602:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12603:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12604:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12605:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12606:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12607:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12608:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12609:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12610:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12611:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12612:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12613:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12614:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12615:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12616:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12617:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12618:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12619:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12620:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12621:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12622:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12623:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12624:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12625:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12626:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12627:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12628:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12629:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12630:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12631:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12632:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12633:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12634:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12635:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12636:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12637:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12638:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12639:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12640:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12641:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12642:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12643:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12644:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
```

```
12645:       var lpLuid: TLargeInteger): BOOL; stdcall;
12646:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12647:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12648:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12649:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12650:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12651:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12652:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12653:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12654:       var GenerateOnClose: BOOL): BOOL; stdcall;
12655:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12656:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12657:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12658:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12659:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12660:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12661:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12662:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12663:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12664:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12665:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12666:       var phkResult: HKEY): Longint; stdcall;
12667:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12668:       var phkResult: HKEY): Longint; stdcall;
12669:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12670:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12671:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12672:       lpdwDisposition: PDWORD): Longint; stdcall;
12673:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12674:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12675:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12676:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12677:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12678:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12679:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12680:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12681:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12682:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12683:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12684:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12685:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12686:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12687:       lpcbClass: PDWORD; lpReserved: Pointer;
12688:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12689:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12690:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12691:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12692:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12693:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12694:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12695:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12696:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12697:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12698:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12699:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12700:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12701:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12702:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12703:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12704:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12705:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12706:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12707:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12708:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12709:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12710:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12711:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12712:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12713:
12714:  Function wAddAtom( lpString : PKOLChar) : ATOM
12715:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12716:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12717:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12718:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12719:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12720:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12721:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12722:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12723:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12724:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12725:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12726:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
```

```
12727:  Function
        CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12728:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
        nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12729:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
        lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
        Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
        lpProcessInfo:TProcessInformation):BOOL
12730:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
        Longint; lpName : PKOLChar) : THandle
12731:  Function
        wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12732:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12733:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12734:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12735:  //Function
        wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12736:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12737:  //Function
        wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12738:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12739:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12740:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12741:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12742:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12743:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12744:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
        dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12745:  Function wFindAtom( lpString : PKOLChar) : ATOM
12746:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12747:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12748:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12749:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12750:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12751:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12752:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12753:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12754:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12755:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12756:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12757:  Function wGetCommandLine : PKOLChar
12758:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12759:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12760:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12761:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12762:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12763:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12764:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12765:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12766:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12767:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12768:  Function wGetEnvironmentStrings : PKOLChar
12769:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12770:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12771:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12772:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12773:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12774:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12775:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12776:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12777:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12778:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12779:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12780:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12781:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12782:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12783:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12784:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12785:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12786:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12787:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12788: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12789:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
```

```
12790:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12791:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12792: //Function
       wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12793: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12794: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
       : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
       lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12795:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12796:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12797:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12798:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12799:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12800:  Function
       wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12801:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12802:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12803:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12804:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12805:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
       TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12806:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12807:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12808:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12809:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12810:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12811:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12812:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
       lpNumberOfEventsRead:DWORD):BOOL;
12813:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12814:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12815:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
       lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12816:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
       lpNumbOfEventsRead:DWORD):BOOL;
12817:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
       : TCoord; var lpReadRegion : TSmallRect) : BOOL
12818:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
       DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12819:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12820:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
       lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12821:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
       lpFilePart:PKOLChar):DWORD;
12822:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12823:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12824:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12825:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12826:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12827:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12828:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12829:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12830:  //Function wUpdateResource(hUpdate:THandle;lpType,
       lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12831:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12832:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12833:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
       DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12834:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
       var lpNumberOfEventsWritten : DWORD) : BOOL
12835:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
       TCoord; var lpWriteRegion : TSmallRect) : BOOL
12836:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
       DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12837:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12838:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12839:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12840:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12841:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12842:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12843:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12844:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12845:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12846:  Function wlstrlen( lpString : PKOLChar) : Integer
12847:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
       PNetConnectInfoStruct) : DWORD
12848:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
       lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12849:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
       lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12850:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12851:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12852:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12853:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12854:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12855:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12856:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12857:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
       : PKOLChar; nNameBufSize : DWORD) : DWORD
```

```
12858:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12859:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12860:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12861: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
       lpBufferSize:DWORD):DWORD;
12862:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12863: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12864: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12865:  //Function wWNetUseConnection(hwndOwner:HWND;var
       lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
       lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12866:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12867:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12868:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
       lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12869:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
       szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12870: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12871: //Func wGetPrivateProfileStruct(lpszSection,
       lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12872: //Func wWritePrivateProfileStruct(lpszSection,
       lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12873:  Function wAddFontResource( FileName : PKOLChar) : Integer
12874:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12875:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12876:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12877:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12878:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12879: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12880:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
       fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
       fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12881:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12882:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12883: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12884:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12885:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12886: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
       pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12887: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12888:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12889:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12890:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12891:  //Function wExtTextOut(DC:HDC;X,
       Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12892:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12893:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12894:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12895:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12896: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12897: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12898:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12899:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12900: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12901: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
       lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12902:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12903: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12904:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12905: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12906:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12907:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12908:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12909:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12910:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12911:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12912:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12913:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12914:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12915:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12916:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12917:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12918:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12919:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12920:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12921:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12922:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12923:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12924:  //Function
       wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12925:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12926: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
       dwFlags : DWORD; lParam : Pointer) : Longint
12927:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12928:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12929:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12930:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12931:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12932:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
```

```
12933: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12934:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12935:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12936:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12937:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12938:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12939:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12940:  //Function wCreateDesktop(lpszDesktop,
       lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12941:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
       HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12942:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
       lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12943:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
       hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12944: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
       nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12945:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
       dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12946:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12947:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12948:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12949:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12950:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
       : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12951:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
       : TFNDlgProc; dwInitParam : LPARAM) : Integer
12952:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12953:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
       nIDStaticPath:Integer;uFileType:UINT):Integer;
12954:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
       nIDStaticPath:Int;uFiletype:UINT):Int;
12955:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12956:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12957: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
       cy:Int;Flags:UINT):BOOL;
12958:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12959:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12960:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12961:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
       pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12962: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12963:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12964:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12965:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12966:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12967:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12968:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12969:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12970:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12971:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12972:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12973:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12974:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
       lpnTabStopPositions) : DWORD
12975: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
       lpnLengthNeed:DWORD)BOOL;
12976:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12977:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12978:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12979:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12980: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
       nHeigt:Int):BOOL;
12981:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12982:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12983:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12984:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12985:  Function wIsCharLower( ch : KOLChar) : BOOL
12986:  Function wIsCharUpper( ch : KOLChar) : BOOL
12987:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12988:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12989:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12990:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12991:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12992:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12993:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12994:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12995:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12996:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12997:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12998:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12999:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
13000:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
13001:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
13002:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
13003:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
13004:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
13005:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
13006:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
```

```
13007:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
13008:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
13009:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
13010:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
13011:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13012:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13013:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
13014: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
13015: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
13016:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
13017: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13018:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
13019:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
13020:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13021:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
13022:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
       lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
13023:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
       lpdwResult:DWORD): LRESULT
13024:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13025:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
13026:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
13027:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13028:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13029: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13030:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
13031:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13032:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
13033:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13034: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
13035:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
       lpnTabStopPositions,nTabOrigin:Int):Longint;
13036:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
13037:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13038:  Function wVkKeyScan( ch : KOLChar) : SHORT
13039:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13040:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
13041:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
13042:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
13043:
13044: //TestDrive!
13045: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13046:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13047:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
13048:  Function GetLocalUserSidStr( const UserName : string) : string
13049:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
13050:  Function Impersonate2User( const domain : string; const user : string) : boolean
13051:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
13052:  Function KillProcessbyname( const exename : string; var found : integer) : integer
13053:  Function getWinProcessList : TStringList
13054:  function WaitTilClose(hWnd: Integer): Integer;
13055:  function DoUserMsgs: Boolean;
13056:  function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13057:  procedure ShowMsg(hParent: Integer; const Mess, Title: String);  //modal but NOT blockable
13058:  procedure DeleteMsgForm(Handle: Integer);
13059:  procedure DisableForms;
13060:  function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13061: end;
13062:
13063: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13064: begin
13065:  'AfMaxSyncSlots','LongInt'( 64);
13066:  'AfSynchronizeTimeout','LongInt'( 2000);
13067:   TAfSyncSlotID', 'DWORD
13068:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13069:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13070:   TAfSafeDirectSyncEvent', 'Procedure
13071:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13072:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13073:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13074:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13075:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13076:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13077:  Function AfIsSyncMethod : Boolean
13078:  Function AfSyncWnd : HWnd
13079:  Function AfSyncStatistics : TAfSyncStatistics
13080:  Procedure AfClearSyncStatistics
13081: end;
13082:
13083: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13084: begin
13085:  'fBinary','LongWord')( $00000001);
13086:  'fParity','LongWord'( $00000002);
13087:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13088:  'fOutxDsrFlow','LongWord')( $00000008);
13089:  'fDtrControl','LongWord')( $00000030);
13090:  'fDtrControlDisable','LongWord')( $00000000);
13091:  'fDtrControlEnable','LongWord')( $00000010);
13092:  'fDtrControlHandshake','LongWord')( $00000020);
```

```
13093:  'fDsrSensitivity','LongWord')( $00000040);
13094:  'fTXContinueOnXoff','LongWord')( $00000080);
13095:  'fOutX','LongWord')( $00000100);
13096:  'fInX','LongWord')( $00000200);
13097:  'fErrorChar','LongWord')( $00000400);
13098:  'fNull','LongWord')( $00000800);
13099:  'fRtsControl','LongWord')( $00003000);
13100:  'fRtsControlDisable','LongWord')( $00000000);
13101:  'fRtsControlEnable','LongWord')( $00001000);
13102:  'fRtsControlHandshake','LongWord')( $00002000);
13103:  'fRtsControlToggle','LongWord')( $00003000);
13104:  'fAbortOnError','LongWord')( $00004000);
13105:  'fDummy2','LongWord')( $FFFF8000);
13106:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13107:   FindClass('TOBJECT'),'EAfComPortCoreError
13108:   FindClass('TOBJECT'),'TAfComPortCore
13109:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13110:    +'tKind : TAfCoreEvent; Data : DWORD)
13111:   SIRegister_TAfComPortCoreThread(CL);
13112:   SIRegister_TAfComPortEventThread(CL);
13113:   SIRegister_TAfComPortWriteThread(CL);
13114:   SIRegister_TAfComPortCore(CL);
13115:  Function FormatDeviceName( PortNumber : Integer) : string
13116: end;
13117:
13118: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13119: begin
13120:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13121:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13122:   SIRegister_TApplicationFileIO(CL);
13123:   TDataFileCapability', '( dfcRead, dfcWrite )
13124:   TDataFileCapabilities', 'set of TDataFileCapability
13125:   SIRegister_TDataFile(CL);
13126:   //TDataFileClass', 'class of TDataFile
13127:  Function ApplicationFileIODefined : Boolean
13128:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13129:  Function FileStreamExists(const fileName: String) : Boolean
13130:  //Procedure Register
13131: end;
13132:
13133: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13134: begin
13135:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13136:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13137:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13138:   TALFBXScale', 'Integer
13139:   FindClass('TOBJECT'),'EALFBXConvertError
13140:   SIRegister_EALFBXError(CL);
13141:   SIRegister_EALFBXException(CL);
13142:   FindClass('TOBJECT'),'EALFBXGFixError
13143:   FindClass('TOBJECT'),'EALFBXDSQLError
13144:   FindClass('TOBJECT'),'EALFBXDynError
13145:   FindClass('TOBJECT'),'EALFBXGBakError
13146:   FindClass('TOBJECT'),'EALFBXGSecError
13147:   FindClass('TOBJECT'),'EALFBXLicenseError
13148:   FindClass('TOBJECT'),'EALFBXGStatError
13149:   //EALFBXExceptionClass', 'class of EALFBXError
13150:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13151:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13152:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13153:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13154:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13155:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13156:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13157:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13158:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13159:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13160:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13161:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13162:   TALFBXTransParams', 'set of TALFBXTransParam
13163:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13164:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13165:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13166: 'cALFBXMaxParamLength','LongInt'( 125);
13167:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13168:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13169:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13170:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13171:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13172:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13173:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13174:   SIRegister_TALFBXSQLDA(CL);
13175:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13176:   SIRegister_TALFBXPoolStream(CL);
13177:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13178:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13179:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13180:   //TALFBXArrayDesc', 'TISCArrayDesc
13181:   //TALFBXBlobDesc', 'TISCBlobDesc
```

```
13182:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13183:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13184:   SIRegister_TALFBXSQLResult(CL);
13185:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13186:   SIRegister_TALFBXSQLParams(CL);
13187:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13188:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13189:    +'atementType : TALFBXStatementType; end
13190:   FindClass('TOBJECT'),'TALFBXLibrary
13191:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13192:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13193:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13194:    //+' Excep : EALFBXExceptionClass)
13195:   SIRegister_TALFBXLibrary(CL);
13196:  'cAlFBXDateOffset','LongInt'( 15018);
13197:  'cALFBXTimeCoeff','LongInt'( 864000000);
13198:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13199:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13200:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13201:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13202:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13203:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13204:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13205:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13206:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13207:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13208:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13209:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13210:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13211:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13212: end;
13213:
13214: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13215: begin
13216:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13217:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13218:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13219:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13220:    +'teger; First : Integer; CacheThreshold : Integer; end
13221:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13222:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13223:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13224:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13225:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13226:   SIRegister_TALFBXClient(CL);
13227:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13228:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13229:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13230:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13231:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13232:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13233:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13234:   SIRegister_TALFBXConnectionPoolClient(CL);
13235:   SIRegister_TALFBXEventThread(CL);
13236:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13237: end;
13238:
13239: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13240: begin
13241: _OSVERSIONINFOA = record
13242:     dwOSVersionInfoSize: DWORD;
13243:     dwMajorVersion: DWORD;
13244:     dwMinorVersion: DWORD;
13245:     dwBuildNumber: DWORD;
13246:     dwPlatformId: DWORD;
13247:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13248:   end;
13249:  TOSVersionInfoA', '_OSVERSIONINFOA
13250:   TOSVersionInfo', 'TOSVersionInfoA
13251:  'WS_EX_RIGHT','LongWord')( $00001000);
13252:  'WS_EX_LEFT','LongWord')( $00000000);
13253:  'WS_EX_RTLREADING','LongWord')( $00002000);
13254:  'WS_EX_LTRREADING','LongWord')( $00000000);
13255:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13256:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13257:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13258:  'LAYOUT_RTL','LongWord')( $00000001);
13259:  'LAYOUT_BTT','LongWord')( $00000002);
13260:  'LAYOUT_VBH','LongWord')( $00000004);
13261:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13262:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13263:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13264:  Function GetLayout( dc : hdc) : DWORD
13265:  Function IsBidi : Boolean
13266:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13267:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13268:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13269:  Function GetPriorityClass( hProcess : THandle) : DWORD
13270:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
```

```
13271:  Function CloseClipboard : BOOL
13272:  Function GetClipboardSequenceNumber : DWORD
13273:  Function GetClipboardOwner : HWND
13274:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13275:  Function GetClipboardViewer : HWND
13276:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13277:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13278:  Function GetClipboardData( uFormat : UINT) : THandle
13279:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13280:  Function CountClipboardFormats : Integer
13281:  Function EnumClipboardFormats( format : UINT) : UINT
13282:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13283:  Function EmptyClipboard : BOOL
13284:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13285:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13286:  Function GetOpenClipboardWindow : HWND
13287:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13288:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13289:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13290:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13291:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13292:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13293:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13294:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13295:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13296: end;
13297:
13298: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13299: begin
13300:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13301:  Function GetTemporaryFilesPath : String
13302:  Function GetTemporaryFileName : String
13303:  Function FindFileInPaths( const fileName, paths : String) : String
13304:  Function PathsToString( const paths : TStrings) : String
13305:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13306:  //Function MacroExpandPath( const aPath : String) : String
13307: end;
13308:
13309: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13310: begin
13311:   SIRegister_TALMultiPartBaseContent(CL);
13312:   SIRegister_TALMultiPartBaseContents(CL);
13313:   SIRegister_TAlMultiPartBaseStream(CL);
13314:   SIRegister_TALMultipartBaseEncoder(CL);
13315:   SIRegister_TALMultipartBaseDecoder(CL);
13316:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13317:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13318:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13319: end;
13320:
13321: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13322: begin
13323:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13324:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13325:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13326:  Function aAllocPadedMem( Size : Cardinal) : TObject
13327:  Procedure aFreePadedMem( var P : TObject);
13328:  Procedure aFreePadedMem1( var P : PChar);
13329:  Function aCheckPadedMem( P : Pointer) : Byte
13330:  Function aGetPadMemSize( P : Pointer) : Cardinal
13331:  Function aAllocMem( Size : Cardinal) : Pointer
13332:  Function aStrLen( const Str : PChar) : Cardinal
13333:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13334:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13335:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13336:  Function aStrEnd( const Str : PChar) : PChar
13337:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13338:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13339:  Function aPCharLength( const Str : PChar) : Cardinal
13340:  Function aPCharUpper( Str : PChar) : PChar
13341:  Function aPCharLower( Str : PChar) : PChar
13342:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13343:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13344:  Function aCopyTail( const S : String; Len : Integer) : String
13345:  Function aInt2Thos( I : Int64) : String
13346:  Function aUpperCase( const S : String) : String
13347:  Function aLowerCase( const S : string) : String
13348:  Function aCompareText( const S1, S2 : string) : Integer
13349:  Function aSameText( const S1, S2 : string) : Boolean
13350:  Function aInt2Str( Value : Int64) : String
13351:  Function aStr2Int( const Value : String) : Int64
13352:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13353:  Function aGetFileExt( const FileName : String) : String
13354:  Function aGetFilePath( const FileName : String) : String
13355:  Function aGetFileName( const FileName : String) : String
13356:  Function aChangeExt( const FileName, Extension : String) : String
13357:  Function aAdjustLineBreaks( const S : string) : string
13358:  Function aGetWindowStr( WinHandle : HWND) : String
13359:  Function aDiskSpace( Drive : String) : TdriveSize
```

```
13360:  Function aFileExists( FileName : String) : Boolean
13361:  Function aFileSize( FileName : String) : Int64
13362:  Function aDirectoryExists( const Name : string) : Boolean
13363:  Function aSysErrorMessage( ErrorCode : Integer) : string
13364:  Function aShortPathName( const LongName : string) : string
13365:  Function aGetWindowVer : TWinVerRec
13366:  procedure InitDriveSpacePtr;
13367: end;
13368:
13369: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13370: begin
13371:  aZero','LongInt'( 0);
13372:  'makeappDEF','LongInt'( - 1);
13373:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13374:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13375:  'CS_KEYCVTWINDOW','LongInt'( 4);
13376:  'CS_DBLCLKS','LongInt'( 8);
13377:  'CS_OWNDC','LongWord')( $20);
13378:  'CS_CLASSDC','LongWord')( $40);
13379:  'CS_PARENTDC','LongWord')( $80);
13380:  'CS_NOKEYCVT','LongWord')( $100);
13381:  'CS_NOCLOSE','LongWord')( $200);
13382:  'CS_SAVEBITS','LongWord')( $800);
13383:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13384:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13385:  'CS_GLOBALCLASS','LongWord')( $4000);
13386:  'CS_IME','LongWord')( $10000);
13387:  'CS_DROPSHADOW','LongWord')( $20000);
13388:   //PPanelFunc', '^TPanelFunc // will not work
13389:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13390:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13391:   TFontLooks', 'set of TFontLook
13392:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13393:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13394:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13395:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13396:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13397:  Procedure RunMsgLoop( Show : Boolean)
13398:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13399:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13400:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13401:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13402:  Function MakePanel(Left,Top,Width,Height,
       hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13403:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13404:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13405:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13406: end;
13407:
13408: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13409: begin
13410:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13411:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13412:   TScreenSaverOptions', 'set of TScreenSaverOption
13413:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13414:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13415:   SIRegister_TScreenSaver(CL);
13416:  //Procedure Register
13417:  Procedure SetScreenSaverPassword
13418: end;
13419:
13420: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13421: begin
13422:   FindClass('TOBJECT'),'TXCollection
13423:   SIRegister_EFilerException(CL);
13424:   SIRegister_TXCollectionItem(CL);
13425:   //TXCollectionItemClass', 'class of TXCollectionItem
13426:   SIRegister_TXCollection(CL);
13427:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13428:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13429:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13430:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13431:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13432:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13433: end;
13434:
13435: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13436: begin
13437:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13438:  Procedure xglMapTexCoordToNull
13439:  Procedure xglMapTexCoordToMain
13440:  Procedure xglMapTexCoordToSecond
13441:  Procedure xglMapTexCoordToDual
13442:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13443:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13444:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13445:  Procedure xglBeginUpdate
```

```
13446:  Procedure xglEndUpdate
13447:  Procedure xglPushState
13448:  Procedure xglPopState
13449:  Procedure xglForbidSecondTextureUnit
13450:  Procedure xglAllowSecondTextureUnit
13451:  Function xglGetBitWiseMapping : Cardinal
13452: end;
13453:
13454: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13455: begin
13456:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13457:   TBaseListOptions', 'set of TBaseListOption
13458:   SIRegister_TBaseList(CL);
13459:   SIRegister_TBaseVectorList(CL);
13460:   SIRegister_TAffineVectorList(CL);
13461:   SIRegister_TVectorList(CL);
13462:   SIRegister_TTexPointList(CL);
13463:   SIRegister_TXIntegerList(CL);
13464:   //PSingleArrayList', '^TSingleArrayList // will not work
13465:   SIRegister_TSingleList(CL);
13466:   SIRegister_TByteList(CL);
13467:   SIRegister_TQuaternionList(CL);
13468:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13469:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13470:  Procedure FastQuickSortLists(startIndex,
      endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13471: end;
13472:
13473: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13474: begin
13475:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13476:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13477:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
      indices:TIntegerList;list:TAffineVectorList);
13478:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
      indices:TIntegerList;list:TAffineVectorList);
13479:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
      normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13480:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13481:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13482:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13483:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13484:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13485:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13486:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13487:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13488:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13489:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
      edgesTriangles : TIntegerList) : TIntegerList
13490:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13491:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
      TPersistentObjectList;
13492:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13493:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
      TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13494: end;
13495:
13496: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13497: begin
13498:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13499:  Procedure FreeMemAndNil( var P : TObject)
13500:  Function PCharOrNil( const S : string) : PChar
13501:   SIRegister_TJclReferenceMemoryStream(CL);
13502:   FindClass('TOBJECT'),'EJclVMTError
13503:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13504:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13505:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13506:   PDynamicIndexList', '^TDynamicIndexList // will not work
13507:   PDynamicAddressList', '^TDynamicAddressList // will not work
13508:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13509:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13510:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13511:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13512:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13513:  Function GetInitTable( AClass : TClass) : PTypeInfo
13514:   PFieldEntry', '^TFieldEntry // will not work}
13515:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13516:  Function JIsClass( Address : Pointer) : Boolean
13517:  Function JIsObject( Address : Pointer) : Boolean
13518:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13519:   TDigitCount', 'Integer
13520:   SIRegister_TJclNumericFormat(CL);
13521:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13522:   TTextHandler', 'Procedure ( const Text : string)
13523: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13524:  Function JExecute(const
      CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13525:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13526:  Function ReadKey : Char   //to and from the DOS console !
```

```
13527:   TModuleHandle', 'HINST
13528:   //TModuleHandle', 'Pointer
13529:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13530:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13531:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13532:  Procedure UnloadModule( var Module : TModuleHandle)
13533:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13534:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13535:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13536:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13537:   FindClass('TOBJECT'),'EJclConversionError
13538:  Function JStrToBoolean( const S : string) : Boolean
13539:  Function JBooleanToStr( B : Boolean) : string
13540:  Function JIntToBool( I : Integer) : Boolean
13541:  Function JBoolToInt( B : Boolean) : Integer
13542:  'ListSeparator','String ';
13543:  'ListSeparator1','String ':
13544:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13545:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13546:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13547:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13548:  Function ListItemCount( const List, Separator : string) : Integer
13549:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13550:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13551:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13552:  Function SystemTObjectInstance : LongWord
13553:  Function IsCompiledWithPackages : Boolean
13554:  Function JJclGUIDToString( const GUID : TGUID) : string
13555:  Function JJclStringToGUID( const S : string) : TGUID
13556:   SIRegister_TJclIntfCriticalSection(CL);
13557:   SIRegister_TJclSimpleLog(CL);
13558:  Procedure InitSimpleLog( const ALogFileName : string)
13559: end;
13560:
13561: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13562: begin
13563:   FindClass('TOBJECT'),'EJclBorRADException
13564:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13565:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13566:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13567:   TJclBorRADToolPath', 'string
13568:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13569:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13570:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13571:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13572:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13573:  BorRADToolRepositoryFormsPage','String 'Forms
13574:  BorRADToolRepositoryProjectsPage','String 'Projects
13575:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13576:  BorRADToolRepositoryObjectType','String 'Type
13577:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13578:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13579:  BorRADToolRepositoryObjectName','String 'Name
13580:  BorRADToolRepositoryObjectPage','String 'Page
13581:  BorRADToolRepositoryObjectIcon','String 'Icon
13582:  BorRADToolRepositoryObjectDescr','String 'Description
13583:  BorRADToolRepositoryObjectAuthor','String 'Author
13584:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13585:  BorRADToolRepositoryObjectDesigner','String 'Designer
13586:  BorRADToolRepositoryDesignerDfm','String 'dfm
13587:  BorRADToolRepositoryDesignerXfm','String 'xfm
13588:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13589:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13590:  SourceExtensionDelphiPackage','String '.dpk
13591:  SourceExtensionBCBPackage','String '.bpk
13592:  SourceExtensionDelphiProject','String '.dpr
13593:  SourceExtensionBCBProject','String '.bpr
13594:  SourceExtensionBDSProject','String '.bdsproj
13595:  SourceExtensionDProject','String '.dproj
13596:  BinaryExtensionPackage','String '.bpl
13597:  BinaryExtensionLibrary','String '.dll
13598:  BinaryExtensionExecutable','String '.exe
13599:  CompilerExtensionDCP','String '.dcp
13600:  CompilerExtensionBPI','String '.bpi
13601:  CompilerExtensionLIB','String '.lib
13602:  CompilerExtensionTDS','String '.tds
13603:  CompilerExtensionMAP','String '.map
13604:  CompilerExtensionDRC','String '.drc
13605:  CompilerExtensionDEF','String '.def
13606:  SourceExtensionCPP','String '.cpp
13607:  SourceExtensionH','String '.h
13608:  SourceExtensionPAS','String '.pas
13609:  SourceExtensionDFM','String '.dfm
13610:  SourceExtensionXFM','String '.xfm
13611:  SourceDescriptionPAS','String 'Pascal source file
13612:  SourceDescriptionCPP','String 'C++ source file
13613:  DesignerVCL','String 'VCL
13614:  DesignerCLX','String 'CLX
13615:  ProjectTypePackage','String 'package
```

```
13616:  ProjectTypeLibrary','String 'library
13617:  ProjectTypeProgram','String 'program
13618:  Personality32Bit','String '32 bit
13619:  Personality64Bit','String '64 bit
13620:  PersonalityDelphi','String 'Delphi
13621:  PersonalityDelphiDotNet','String 'Delphi.net
13622:  PersonalityBCB','String 'C++Builder
13623:  PersonalityCSB','String 'C#Builder
13624:  PersonalityVB','String 'Visual Basic
13625:  PersonalityDesign','String 'Design
13626:  PersonalityUnknown','String 'Unknown personality
13627:  PersonalityBDS','String 'Borland Developer Studio
13628:  DOFDirectoriesSection','String 'Directories
13629:  DOFUnitOutputDirKey','String 'UnitOutputDir
13630:  DOFSearchPathName','String 'SearchPath
13631:  DOFConditionals','String 'Conditionals
13632:  DOFLinkerSection','String 'Linker
13633:  DOFPackagesKey','String 'Packages
13634:  DOFCompilerSection','String 'Compiler
13635:  DOFPackageNoLinkKey','String 'PackageNoLink
13636:  DOFAdditionalSection','String 'Additional
13637:  DOFOptionsKey','String 'Options
13638:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13639:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13640:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13641:   TJclBorPersonalities', 'set of TJclBorPersonality
13642:   TJclBorDesigner', '( bdVCL, bdCLX )
13643:   TJclBorDesigners', 'set of TJClBorDesigner
13644:   TJclBorPlatform', '( bp32bit, bp64bit )
13645:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13646:   SIRegister_TJclBorRADToolInstallationObject(CL);
13647:   SIRegister_TJclBorlandOpenHelp(CL);
13648:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13649:   TJclHelp2Objects', 'set of TJclHelp2Object
13650:   SIRegister_TJclHelp2Manager(CL);
13651:   SIRegister_TJclBorRADToolIdeTool(CL);
13652:   SIRegister_TJclBorRADToolIdePackages(CL);
13653:   SIRegister_IJclCommandLineTool(CL);
13654:   FindClass('TOBJECT'),'EJclCommandLineToolError
13655:   SIRegister_TJclCommandLineTool(CL);
13656:   SIRegister_TJclBorlandCommandLineTool(CL);
13657:   SIRegister_TJclBCC32(CL);
13658:   SIRegister_TJclDCC32(CL);
13659:   TJclDCC', 'TJclDCC32
13660:   SIRegister_TJclBpr2Mak(CL);
13661:   SIRegister_TJclBorlandMake(CL);
13662:   SIRegister_TJclBorRADToolPalette(CL);
13663:   SIRegister_TJclBorRADToolRepository(CL);
13664:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13665:   TCommandLineTools', 'set of TCommandLineTool
13666:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13667:   SIRegister_TJclBorRADToolInstallation(CL);
13668:   SIRegister_TJclBCBInstallation(CL);
13669:   SIRegister_TJclDelphiInstallation(CL);
13670:   SIRegister_TJclDCCIL(CL);
13671:   SIRegister_TJclBDSInstallation(CL);
13672:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13673:   SIRegister_TJclBorRADToolInstallations(CL);
13674:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13675:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13676:  Function IsDelphiPackage( const FileName : string) : Boolean
13677:  Function IsDelphiProject( const FileName : string) : Boolean
13678:  Function IsBCBPackage( const FileName : string) : Boolean
13679:  Function IsBCBProject( const FileName : string) : Boolean
13680:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13681:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13682:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13683:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
13684:  function SamePath(const Path1, Path2: string): Boolean;
13685: end;
13686:
13687: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13688: begin
13689:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13690:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13691:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13692:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13693:  'LPathSeparator','String '/
13694:  'LDirDelimiter','String '/
13695:  'LDirSeparator','String ':
13696:  'JXPathDevicePrefix','String '\\.\
13697:  'JXPathSeparator','String '\
13698:  'JXDirDelimiter','String '\
13699:  'JXDirSeparator','String ';
13700:  'JXPathUncPrefix','String '\\
13701:  'faNormalFile','LongWord')( $00000080);
13702:  //'faUnixSpecific',' faSymLink);
```

```
13703:   JXTCompactPath', '( cpCenter, cpEnd )
13704:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13705:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13706:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13707:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13708:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13709:
13710:  Function jxPathAddSeparator( const Path : string) : string
13711:  Function jxPathAddExtension( const Path, Extension : string) : string
13712:  Function jxPathAppend( const Path, Append : string) : string
13713:  Function jxPathBuildRoot( const Drive : Byte) : string
13714:  Function jxPathCanonicalize( const Path : string) : string
13715:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13716:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13717:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13718:  Function jxPathExtractFileDirFixed( const S : string) : string
13719:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13720:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13721:  Function jxPathGetDepth( const Path : string) : Integer
13722:  Function jxPathGetLongName( const Path : string) : string
13723:  Function jxPathGetShortName( const Path : string) : string
13724:  Function jxPathGetLongName( const Path : string) : string
13725:  Function jxPathGetShortName( const Path : string) : string
13726:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13727:  Function jxPathGetTempPath : string
13728:  Function jxPathIsAbsolute( const Path : string) : Boolean
13729:  Function jxPathIsChild( const Path, Base : string) : Boolean
13730:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13731:  Function jxPathIsUNC( const Path : string) : Boolean
13732:  Function jxPathRemoveSeparator( const Path : string) : string
13733:  Function jxPathRemoveExtension( const Path : string) : string
13734:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13735:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13736:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13737:   JxTFileListOptions', 'set of TFileListOption
13738:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13739:   TFileHandler', 'Procedure ( const FileName : string)
13740:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13741:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13742:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
      AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
      FileMatchFunc:TFileMatchFunc):Bool;
13743:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13744:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13745:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13746:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13747:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
      RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13748:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
      IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13749:  Procedure jxCreateEmptyFile( const FileName : string)
13750:  Function jxCloseVolume( var Volume : THandle) : Boolean
13751:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13752:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13753:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13754:  Function jxDelTree( const Path : string) : Boolean
13755:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13756:  Function jxDiskInDrive( Drive : Char) : Boolean
13757:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13758:  Function jxFileCreateTemp( var Prefix : string) : THandle
13759:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13760:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13761:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13762:  Function jxFileExists( const FileName : string) : Boolean
13763:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13764:  Function jxFileRestore( const FileName : string) : Boolean
13765:  Function jxGetBackupFileName( const FileName : string) : string
13766:  Function jxIsBackupFileName( const FileName : string) : Boolean
13767:  Function jxFileGetDisplayName( const FileName : string) : string
13768:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13769:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13770:  Function jxFileGetSize( const FileName : string) : Int64
13771:  Function jxFileGetTempName( const Prefix : string) : string
13772:  Function jxFileGetTypeName( const FileName : string) : string
13773:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13774:  Function jxForceDirectories( Name : string) : Boolean
13775:  Function jxGetDirectorySize( const Path : string) : Int64
13776:  Function jxGetDriveTypeStr( const Drive : Char) : string
13777:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13778:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13779:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13780:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13781:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13782:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
      ResolveSymLinks:Boolean):Integer
13783:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13784:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13785:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13786:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
```

```
13787:  Function jxGetFileCreation( const FName : string) : TFileTime;
13788:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13789:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13790:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13791:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13792:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13793:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13794:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13795:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13796:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13797:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13798:  Function jxGetModulePath( const Module : HMODULE) : string
13799:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13800:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13801:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13802:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13803:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13804:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13805:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13806:  Function jxOpenVolume( const Drive : Char) : THandle
13807:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13808:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13809:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13810:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13811:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13812:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13813:  Procedure jxShredFile( const FileName : string; Times : Integer)
13814:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13815:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13816:  Function jxSymbolicLinkTarget( const Name : string) : string
13817:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13818:   SIRegister_TJclCustomFileAttrMask(CL);
13819:   SIRegister_TJclFileAttributeMask(CL);
13820:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13821:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13822:   TFileSearchOptions', 'set of TFileSearchOption
13823:   TFileSearchTaskID', 'Integer
13824:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13825:    +'hTaskID; const Aborted : Boolean)
13826:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13827:   SIRegister_IJclFileEnumerator(CL);
13828:   SIRegister_TJclFileEnumerator(CL);
13829:  Function JxFileSearch : IJclFileEnumerator
13830:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13831:   JxTFileFlags', 'set of TFileFlag
13832:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13833:   SIRegister_TJclFileVersionInfo(CL);
13834:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13835:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13836:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13837:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13838:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13839:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13840:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13841:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13842:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13843:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13844:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13845:   SIRegister_TJclTempFileStream(CL);
13846:   FindClass('TOBJECT'),'TJclCustomFileMapping
13847:   SIRegister_TJclFileMappingView(CL);
13848:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13849:   SIRegister_TJclCustomFileMapping(CL);
13850:   SIRegister_TJclFileMapping(CL);
13851:   SIRegister_TJclSwapFileMapping(CL);
13852:   SIRegister_TJclFileMappingStream(CL);
13853:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13854:   //PPCharArray', '^TPCharArray // will not work
13855:   SIRegister_TJclMappedTextReader(CL);
13856:   SIRegister_TJclFileMaskComparator(CL);
13857:   FindClass('TOBJECT'),'EJclPathError
13858:   FindClass('TOBJECT'),'EJclFileUtilsError
13859:   FindClass('TOBJECT'),'EJclTempFileStreamError
13860:   FindClass('TOBJECT'),'EJclTempFileStreamError
13861:   FindClass('TOBJECT'),'EJclFileMappingError
13862:   FindClass('TOBJECT'),'EJclFileMappingViewError
13863:  Function jxPathGetLongName2( const Path : string) : string
13864:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13865:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13866:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13867:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13868:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13869:  Procedure jxPathListAddItems( var List : string; const Items : string)
13870:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13871:  Procedure jxPathListDelItems( var List : string; const Items : string)
13872:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13873:  Function jxPathListItemCount( const List : string) : Integer
```

```
13874:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13875:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13876:  Function jxPathListItemIndex( const List, Item : string) : Integer
13877:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13878:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13879:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13880:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string) : Integer
13881: end;
13882:
13883: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13884: begin
13885:  'UTF8FileHeader','String #$ef#$bb#$bf);
13886:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13887:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13888:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13889:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13890:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13891:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13892:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13893:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13894:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13895:  Function lFileIsReadable( const AFilename : string) : boolean
13896:  Function lFileIsWritable( const AFilename : string) : boolean
13897:  Function lFileIsText( const AFilename : string) : boolean
13898:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13899:  Function lFileIsExecutable( const AFilename : string) : boolean
13900:  Function lFileIsSymlink( const AFilename : string) : boolean
13901:  Function lFileIsHardLink( const AFilename : string) : boolean
13902:  Function lFileSize( const Filename : string) : int64;
13903:  Function lGetFileDescription( const AFilename : string) : string
13904:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13905:  Function lTryReadAllLinks( const Filename : string) : string
13906:  Function lDirPathExists( const FileName : String) : Boolean
13907:  Function lForceDirectory( DirectoryName : string) : boolean
13908:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13909:  Function lProgramDirectory : string
13910:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13911:  Function lExtractFileNameOnly( const AFilename : string) : string
13912:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13913:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13914:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13915:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13916:  Function lAppendPathDelim( const Path : string) : string
13917:  Function lChompPathDelim( const Path : string) : string
13918:  Function lTrimFilename( const AFilename : string) : string
13919:  Function lCleanAndExpandFilename( const Filename : string) : string
13920:  Function lCleanAndExpandDirectory( const Filename : string) : string
13921:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13922:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13923:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13924:  Function lFileIsInPath( const Filename, Path : string) : boolean
13925:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13926:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13927:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13928:  'AllDirectoryEntriesMask','String '*
13929:  Function lGetAllFilesMask : string
13930:  Function lGetExeExt : string
13931:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13932:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13933:  Function lFindDiskFilename( const Filename : string) : string
13934:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13935:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13936:  Function lGetDarwinSystemFilename( Filename : string) : string
13937:   SIRegister_TFileIterator(CL);
13938:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13939:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13940:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13941:   SIRegister_TFileSearcher(CL);
13942:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13943:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13944:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13945:  // TCopyFileFlags', 'set of TCopyFileFlag
13946:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13947:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13948:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13949:  Function lReadFileToString( const Filename : string) : string
13950:  Function lGetTempFilename( const Directory, Prefix : string) : string
13951:  {Function NeedRTLAnsi : boolean
13952:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13953:  Function UTF8ToSys( const s : string) : string
13954:  Function SysToUTF8( const s : string) : string
13955:  Function ConsoleToUTF8( const s : string) : string
13956:  Function UTF8ToConsole( const s : string) : string}
13957:  Function FileExistsUTF8( const Filename : string) : boolean
```

```
13958:  Function FileAgeUTF8( const FileName : string) : Longint
13959:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13960:  Function ExpandFileNameUTF8( const FileName : string) : string
13961:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13962:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13963:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13964:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13965:  Procedure FindCloseUTF8( var F : TSearchrec)
13966:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13967:  Function FileGetAttrUTF8( const FileName : String) : Longint
13968:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13969:  Function DeleteFileUTF8( const FileName : String) : Boolean
13970:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13971:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13972:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13973:  Function GetCurrentDirUTF8 : String
13974:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13975:  Function CreateDirUTF8( const NewDir : String) : Boolean
13976:  Function RemoveDirUTF8( const Dir : String) : Boolean
13977:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13978:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13979:  Function FileCreateUTF8( const FileName : string) : THandle;
13980:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13981:  Function ParamStrUTF8( Param : Integer) : string
13982:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13983:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13984:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13985:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13986:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13987: end;
13988:
13989: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13990: begin
13991:   //VK_F23 = 134;
13992:   //{$EXTERNALSYM VK_F24}
13993:   //VK_F24 = 135;
13994:  TVirtualKeyCode', 'Integer
13995:  'VK_MOUSEWHEELUP','integer'(134);
13996:  'VK_MOUSEWHEELDOWN','integer'(135);
13997:  Function glIsKeyDown( c : Char) : Boolean;
13998:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13999:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
14000:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
14001:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
14002:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
14003:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
14004: end;
14005:
14006: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14007: begin
14008:   TGLPoint', 'TPoint
14009:   //PGLPoint', '^TGLPoint // will not work
14010:   TGLRect', 'TRect
14011:   //PGLRect', '^TGLRect // will not work
14012:   TDelphiColor', 'TColor
14013:   TGLPicture', 'TPicture
14014:   TGLGraphic', 'TGraphic
14015:   TGLBitmap', 'TBitmap
14016:   //TGraphicClass', 'class of TGraphic
14017:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14018:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14019:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14020:    +'Button; Shift : TShiftState; X, Y : Integer)
14021:   TGLMouseMoveEvent', 'TMouseMoveEvent
14022:   TGLKeyEvent', 'TKeyEvent
14023:   TGLKeyPressEvent', 'TKeyPressEvent
14024:   EGLOSError', 'EWin32Error
14025:   EGLOSError', 'EWin32Error
14026:   EGLOSError', 'EOSError
14027:  'glsAllFilter','string'All // sAllFilter
14028:  Function GLPoint( const x, y : Integer) : TGLPoint
14029:  Function GLRGB( const r, g, b : Byte) : TColor
14030:  Function GLColorToRGB( color : TColor) : TColor
14031:  Function GLGetRValue( rgb : DWORD) : Byte
14032:  Function GLGetGValue( rgb : DWORD) : Byte
14033:  Function GLGetBValue( rgb : DWORD) : Byte
14034:  Procedure GLInitWinColors
14035:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
14036:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
14037:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14038:  Procedure GLInformationDlg( const msg : String)
14039:  Function GLQuestionDlg( const msg : String) : Boolean
14040:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14041:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14042:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14043:  Function GLApplicationTerminated : Boolean
14044:  Procedure GLRaiseLastOSError
14045:  Procedure GLFreeAndNil( var anObject: TObject)
14046:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
```

```
14047:  Function GLGetCurrentColorDepth : Integer
14048:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14049:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14050:  Procedure GLSleep( length : Cardinal)
14051:  Procedure GLQueryPerformanceCounter( var val : Int64)
14052:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14053:  Function GLStartPrecisionTimer : Int64
14054:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14055:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14056:  Function GLRDTSC : Int64
14057:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14058:  Function GLOKMessageBox( const Text, Caption : string) : Integer
14059:  Procedure GLShowHTMLUrl( Url : String)
14060:  Procedure GLShowCursor( AShow : boolean)
14061:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14062:  Procedure GLGetCursorPos( var point : TGLPoint)
14063:  Function GLGetScreenWidth : integer
14064:  Function GLGetScreenHeight : integer
14065:  Function GLGetTickCount : int64
14066:  function RemoveSpaces(const str : String) : String;
14067:   TNormalMapSpace','( nmsObject, nmsTangent )
14068:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14069:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14070:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14071:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
14072:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14073: end;
14074:
14075: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14076: begin
14077:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14078:  // PGLStarRecord', '^TGLStarRecord // will not work
14079:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14080:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14081:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14082: end;
14083:
14084:
14085: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14086: begin
14087:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14088:   //PAABB', '^TAABB // will not work
14089:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14090:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14091:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14092:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14093:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14094:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14095:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14096:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14097:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14098:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14099:  Function BBMinX( const c : THmgBoundingBox) : Single
14100:  Function BBMaxX( const c : THmgBoundingBox) : Single
14101:  Function BBMinY( const c : THmgBoundingBox) : Single
14102:  Function BBMaxY( const c : THmgBoundingBox) : Single
14103:  Function BBMinZ( const c : THmgBoundingBox) : Single
14104:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14105:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14106:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14107:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14108:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14109:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14110:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14111:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14112:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14113:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14114:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14115:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14116:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14117:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14118:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14119:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14120:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14121:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14122:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14123:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14124:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14125:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14126:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14127:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14128:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14129:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14130:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14131:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
```

```
14132:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14133:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14134:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14135:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14136:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14137:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
      viewportSizeY:Int):TClipRect
14138: end;
14139:
14140: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14141: begin
14142:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14143:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14144:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14145:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14146:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14147:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14148:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14149:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14150:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14151:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14152:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14153:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14154:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14155:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14156:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14157:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14158:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14159:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14160:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14161:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14162:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14163:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14164:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14165:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14166:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14167: end;
14168:
14169: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14170: begin
14171:  'EPSILON','Single').setExtended( 1e-40);
14172:  'EPSILON2','Single').setExtended( 1e-30);  }
14173:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14174:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14175:  THmgPlane', 'TVector
14176:   TDoubleHmgPlane', 'THomogeneousDblVector
14177:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14178:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14179:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14180:   TSingleArray', 'array of Single
14181:   TTransformations','array [0..15] of Single)
14182:   TPackedRotationMatrix','array [0..2] of Smallint)
14183:   TVertex', 'TAffineVector
14184:   //TVectorGL', 'THomogeneousFltVector
14185:   //TMatrixGL', 'THomogeneousFltMatrix
14186:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14187:  Function glTexPointMake( const s, t : Single) : TTexPoint
14188:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14189:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14190:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14191:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14192:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14193:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14194:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14195:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14196:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14197:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14198:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14199:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14200:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14201:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14202:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14203:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14204:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14205:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14206:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14207:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14208:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14209:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14210:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14211:  Procedure glRstVector( var v : TAffineVector);
14212:  Procedure glRstVector1( var v : TVectorGL);
14213:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14214:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14215:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14216:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14217:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14218:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14219:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
```

```
14220:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14221:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14222:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14223:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14224:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14225:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
        nb:Int;dest:PTexPointArray);
14226:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
        nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14227:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
        PAffineVectorArray);
14228:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14229:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14230:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14231:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14232:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14233:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14234:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14235:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14236:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14237:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14238:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14239:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14240:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14241:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14242:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14243:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14244:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14245:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14246:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14247:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14248:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14249:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14250:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14251:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14252:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14253:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14254:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14255:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14256:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14257:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14258:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14259:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14260:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14261:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14262:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14263:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14264:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14265:  Function glLerp( const start, stop, t : Single) : Single
14266:  Function glAngleLerp( start, stop, t : Single) : Single
14267:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14268:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14269:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14270:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14271:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14272:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14273:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14274:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14275: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14276: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
        PAffineVectorArray);
14277:  Function glVectorLength( const x, y : Single) : Single;
14278:  Function glVectorLength1( const x, y, z : Single) : Single;
14279:  Function glVectorLength2( const v : TAffineVector) : Single;
14280:  Function glVectorLength3( const v : TVectorGL) : Single;
14281:  Function glVectorLength4( const v : array of Single) : Single;
14282:  Function glVectorNorm( const x, y : Single) : Single;
14283:  Function glVectorNorm1( const v : TAffineVector) : Single;
14284:  Function glVectorNorm2( const v : TVectorGL) : Single;
14285:  Function glVectorNorm3( var V : array of Single) : Single;
14286:  Procedure glNormalizeVector( var v : TAffineVector);
14287:  Procedure glNormalizeVector1( var v : TVectorGL);
14288:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14289:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14290: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14291:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14292:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14293:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14294:  Procedure glNegateVector( var V : TAffineVector);
14295:  Procedure glNegateVector2( var V : TVectorGL);
14296:  Procedure glNegateVector3( var V : array of Single);
14297:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14298:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14299:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14300:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14301:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14302:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14303:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14304:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
```

```
14305:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14306:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14307:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14308:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14309:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14310:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14311:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14312:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14313:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14314:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14315:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14316:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14317:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14318:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14319:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14320:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14321:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14322:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14323:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14324:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14325:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14326:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14327:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14328:  Procedure glAbsVector( var v : TVectorGL);
14329:  Procedure glAbsVector1( var v : TAffineVector);
14330:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14331:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14332:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14333:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14334:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14335:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14336:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14337:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14338:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14339:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14340:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14341:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14342:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14343:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14344:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14345:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14346:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14347:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14348:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14349:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14350:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14351:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14352:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14353:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14354:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14355:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14356:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14357:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14358:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14359:  Procedure glAdjointMatrix( var M : TMatrixGL);
14360:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14361:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14362:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14363:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14364:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14365:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14366:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14367:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14368:  Procedure glInvertMatrix( var M : TMatrixGL);
14369:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14370:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14371:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14372:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14373:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14374:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14375:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14376:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14377:  Procedure glNormalizePlane( var plane : THmgPlane)
14378:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14379:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14380:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14381:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14382:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14383:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14384:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14385:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14386:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14387:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14388:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14389:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14390:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14391:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14392:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
```

```
14393:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14394:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14395:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14396:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14397:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14398:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14399:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14400:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14401:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14402:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14403:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14404:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14405:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14406:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14407:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14408:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14409:  Function glLnXP1( X : Extended) : Extended
14410:  Function glLog10( X : Extended) : Extended
14411:  Function glLog2( X : Extended) : Extended;
14412:  Function glLog21( X : Single) : Single;
14413:  Function glLogN( Base, X : Extended) : Extended
14414:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14415:  Function glPower( const Base, Exponent : Single) : Single;
14416:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14417:  Function glDegToRad( const Degrees : Extended) : Extended;
14418:  Function glDegToRad1( const Degrees : Single) : Single;
14419:  Function glRadToDeg( const Radians : Extended) : Extended;
14420:  Function glRadToDeg1( const Radians : Single) : Single;
14421:  Function glNormalizeAngle( angle : Single) : Single
14422:  Function glNormalizeDegAngle( angle : Single) : Single
14423:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14424:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14425:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14426:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14427:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14428:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14429:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14430:  Function glArcCos( const X : Extended) : Extended;
14431:  Function glArcCos1( const x : Single) : Single;
14432:  Function glArcSin( const X : Extended) : Extended;
14433:  Function glArcSin1( const X : Single) : Single;
14434:  Function glArcTan21( const Y, X : Extended) : Extended;
14435:  Function glArcTan21( const Y, X : Single) : Single;
14436:  Function glFastArcTan2( y, x : Single) : Single
14437:  Function glTan( const X : Extended) : Extended;
14438:  Function glTan1( const X : Single) : Single;
14439:  Function glCoTan( const X : Extended) : Extended;
14440:  Function glCoTan1( const X : Single) : Single;
14441:  Function glSinh( const x : Single) : Single;
14442:  Function glSinh1( const x : Double) : Double;
14443:  Function glCosh( const x : Single) : Single;
14444:  Function glCosh1( const x : Double) : Double;
14445:  Function glRSqrt( v : Single) : Single
14446:  Function glRLength( x, y : Single) : Single
14447:  Function glISqrt( i : Integer) : Integer
14448:  Function glILength( x, y : Integer) : Integer;
14449:  Function glILength1( x, y, z : Integer) : Integer;
14450:  Procedure glRegisterBasedExp
14451:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14452:  Function glRoundInt( v : Single) : Single;
14453:  Function glRoundInt1( v : Extended) : Extended;
14454:  Function glTrunc( v : Single) : Integer;
14455:  Function glTrunc64( v : Extended) : Int64;
14456:  Function glInt( v : Single) : Single;
14457:  Function glInt1( v : Extended) : Extended;
14458:  Function glFrac( v : Single) : Single;
14459:  Function glFrac1( v : Extended) : Extended;
14460:  Function glRound( v : Single) : Integer;
14461:  Function glRound64( v : Single) : Int64;
14462:  Function glRound641( v : Extended) : Int64;
14463:  Function glTrunc( X : Extended) : Int64
14464:  Function glRound( X : Extended) : Int64
14465:  Function glFrac( X : Extended) : Extended
14466:  Function glCeil( v : Single) : Integer;
14467:  Function glCeil64( v : Extended) : Int64;
14468:  Function glFloor( v : Single) : Integer;
14469:  Function glFloor64( v : Extended) : Int64;
14470:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14471:  Function glSign( x : Single) : Integer
14472:  Function glIsInRange( const x, a, b : Single) : Boolean;
14473:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14474:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14475:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14476:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14477:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14478:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14479:  Function glMinFloat3( const v1, v2 : Single) : Single;
14480:  Function glMinFloat4( const v : array of Single) : Single;
14481:  Function glMinFloat5( const v1, v2 : Double) : Double;
```

```
14482:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14483:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14484:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14485:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14486:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14487:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14488:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14489:  Function glMaxFloat2( const v : array of Single) : Single;
14490:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14491:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14492:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14493:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14494:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14495:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14496:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14497:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14498:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14499:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14500:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14501:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14502:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14503:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14504:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14505:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14506:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14507:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14508:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14509:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14510:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14511:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14512:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14513:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14514:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14515:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14516:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14517:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14518:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14519:  Procedure glSortArrayAscending( var a : array of Extended)
14520:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14521:  Function glClampValue1( const aValue, aMin : Single) : Single;
14522:  Function glGeometryOptimizationMode : String
14523:  Procedure glBeginFPUOnlySection
14524:  Procedure glEndFPUOnlySection
14525:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14526:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14527:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14528:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14529:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14530:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14531:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14532:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14533:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14534:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14535:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14536:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14537:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14538:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14539:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14540:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14541:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14542:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14543:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14544:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14545:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14546:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14547:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14548:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14549:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14550:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14551:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14552:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14553:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14554:  'cPI','Single').setExtended( 3.141592654);
14555:  'cPIdiv180','Single').setExtended( 0.017453292);
14556:  'c180divPI','Single').setExtended( 57.29577951);
14557:  'c2PI','Single').setExtended( 6.283185307);
14558:  'cPIdiv2','Single').setExtended( 1.570796326);
14559:  'cPIdiv4','Single').setExtended( 0.785398163);
14560:  'c3PIdiv4','Single').setExtended( 2.35619449);
14561:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14562:  'cInv360','Single').setExtended( 1 / 360);
14563:  'c180','Single').setExtended( 180);
14564:  'c360','Single').setExtended( 360);
14565:  'cOneHalf','Single').setExtended( 0.5);
```

```
14566:  'cLn10','Single').setExtended( 2.302585093);
14567:  {'MinSingle','Extended').setExtended( 1.5e-45);
14568:  'MaxSingle','Extended').setExtended( 3.4e+38);
14569:  'MinDouble','Extended').setExtended( 5.0e-324);
14570:  'MaxDouble','Extended').setExtended( 1.7e+308);
14571:  'MinExtended','Extended').setExtended( 3.4e-4932);
14572:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14573:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14574:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14575: end;
14576:
14577: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14578: begin
14579:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14580:   (FindClass('TOBJECT'),'TFaceGroups
14581:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14582:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14583:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14584:   SIRegister_TBaseMeshObject(CL);
14585:   (FindClass('TOBJECT'),'TSkeletonFrameList
14586:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14587:   SIRegister_TSkeletonFrame(CL);
14588:   SIRegister_TSkeletonFrameList(CL);
14589:   (FindClass('TOBJECT'),'TSkeleton
14590:   (FindClass('TOBJECT'),'TSkeletonBone
14591:   SIRegister_TSkeletonBoneList(CL);
14592:   SIRegister_TSkeletonRootBoneList(CL);
14593:   SIRegister_TSkeletonBone(CL);
14594:   (FindClass('TOBJECT'),'TSkeletonColliderList
14595:   SIRegister_TSkeletonCollider(CL);
14596:   SIRegister_TSkeletonColliderList(CL);
14597:   (FindClass('TOBJECT'),'TGLBaseMesh
14598:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14599:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14600:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14601:    +'QuaternionList; end
14602:   SIRegister_TSkeleton(CL);
14603:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14604:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14605:   SIRegister_TMeshObject(CL);
14606:   SIRegister_TMeshObjectList(CL);
14607:   //TMeshObjectListClass', 'class of TMeshObjectList
14608:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14609:   SIRegister_TMeshMorphTarget(CL);
14610:   SIRegister_TMeshMorphTargetList(CL);
14611:   SIRegister_TMorphableMeshObject(CL);
14612:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14613:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14614:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14615:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14616:   SIRegister_TSkeletonMeshObject(CL);
14617:   SIRegister_TFaceGroup(CL);
14618:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14619:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14620:   SIRegister_TFGVertexIndexList(CL);
14621:   SIRegister_TFGVertexNormalTexIndexList(CL);
14622:   SIRegister_TFGIndexTexCoordList(CL);
14623:   SIRegister_TFaceGroups(CL);
14624:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14625:   SIRegister_TVectorFile(CL);
14626:   //TVectorFileClass', 'class of TVectorFile
14627:   SIRegister_TGLGLSMVectorFile(CL);
14628:   SIRegister_TGLBaseMesh(CL);
14629:   SIRegister_TGLFreeForm(CL);
14630:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14631:   TGLActorOptions', 'set of TGLActorOption
14632:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14633:   (FindClass('TOBJECT'),'TGLActor
14634:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14635:   SIRegister_TActorAnimation(CL);
14636:   TActorAnimationName', 'String
14637:   SIRegister_TActorAnimations(CL);
14638:   SIRegister_TGLBaseAnimationControler(CL);
14639:   SIRegister_TGLAnimationControler(CL);
14640:   TActorFrameInterpolation', '( afpNone, afpLinear )
14641:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14642:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14643:   SIRegister_TGLActor(CL);
14644:   SIRegister_TVectorFileFormat(CL);
14645:   SIRegister_TVectorFileFormatsList(CL);
14646:   (FindClass('TOBJECT'),'EInvalidVectorFile
14647:  Function GetVectorFileFormats : TVectorFileFormatsList
14648:  Function VectorFileFormatsFilter : String
14649:  Function VectorFileFormatsSaveFilter : String
14650:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14651:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14652:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14653: end;
14654:
```

```
14655: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14656: begin
14657:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14658:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14659:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14660:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14661:   SIRegister_TOleStream(CL);
14662:   (FindClass('TOBJECT'),'TConnectionPoints
14663:   TConnectionKind', '( ckSingle, ckMulti )
14664:   SIRegister_TConnectionPoint(CL);
14665:   SIRegister_TConnectionPoints(CL);
14666:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14667:   (FindClass('TOBJECT'),'TActiveXControlFactory
14668:   SIRegister_TActiveXControl(CL);
14669:   //TActiveXControlClass', 'class of TActiveXControl
14670:   SIRegister_TActiveXControlFactory(CL);
14671:   SIRegister_TActiveFormControl(CL);
14672:   SIRegister_TActiveForm(CL);
14673:   //TActiveFormClass', 'class of TActiveForm
14674:   SIRegister_TActiveFormFactory(CL);
14675:   (FindClass('TOBJECT'),'TPropertyPageImpl
14676:   SIRegister_TPropertyPage(CL);
14677:   //TPropertyPageClass', 'class of TPropertyPage
14678:   SIRegister_TPropertyPageImpl(CL);
14679:   SIRegister_TActiveXPropertyPage(CL);
14680:   SIRegister_TActiveXPropertyPageFactory(CL);
14681:   SIRegister_TCustomAdapter(CL);
14682:   SIRegister_TAdapterNotifier(CL);
14683:   SIRegister_IFontAccess(CL);
14684:   SIRegister_TFontAdapter(CL);
14685:   SIRegister_IPictureAccess(CL);
14686:   SIRegister_TPictureAdapter(CL);
14687:   SIRegister_TOleGraphic(CL);
14688:   SIRegister_TStringsAdapter(CL);
14689:   SIRegister_TReflectorWindow(CL);
14690:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14691:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14692:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14693:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14694:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14695:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14696:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14697:  Function ParkingWindow : HWND
14698: end;
14699:
14700: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14701: begin
14702:  // TIp6Bytes = array [0..15] of Byte;
14703: {:binary form of IPv6 adress (for string conversion routines)}
14704:  // TIp6Words = array [0..7] of Word;
14705:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14706:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14707:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14708:  Function synaIsIP( const Value : string) : Boolean');
14709:  Function synaIsIP6( const Value : string) : Boolean');
14710:  Function synaIPToID( Host : string) : Ansistring');
14711:  Function synaStrToIp6( value : string) : TIp6Bytes');
14712:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14713:  Function synaStrToIp( value : string) : integer');
14714:  Function synaIpToStr( value : integer) : string');
14715:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14716:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14717:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14718:  Function xStrToIP( const Value : String) : TIPAdr');
14719:  Function xIPToStr( const Adresse : TIPAdr) : String');
14720:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14721:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14722: end;
14723:
14724: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14725: begin
14726:  AddTypeS('TSpecials', 'set of Char');
14727:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14728:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14729:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14730:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14731:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14732:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14733:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14734:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14735:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14736:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14737:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14738:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14739:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14740:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14741:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14742:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14743:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
```

```
14744:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14745:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14746:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14747:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14748:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14749:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14750:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14751:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14752:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14753:  Function synCrc32( const Value : AnsiString) : Integer');
14754:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14755:  Function Crc16( const Value : AnsiString) : Word');
14756:  Function synMD5( const Value : AnsiString) : AnsiString');
14757:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14758:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14759:  Function synSHA1( const Value : AnsiString) : AnsiString');
14760:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14761:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14762:  Function synMD4( const Value : AnsiString) : AnsiString');
14763: end;
14764:
14765: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14766: begin
14767:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14768:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14769:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14770:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14771:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14772:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14773:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14774:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14775:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14776:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14777:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14778:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14779:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14780:    +', CP864, CP865, CP869, CP1125 )');
14781:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14782:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14783:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
      TransformTable : array of Word) : AnsiString');
14784:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
      TransformTable : array of Word; Translit : Boolean) : AnsiString');
14785:  Function GetCurCP : TMimeChar');
14786:  Function GetCurOEMCP : TMimeChar');
14787:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14788:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14789:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14790:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14791:  Function GetBOM( Value : TMimeChar) : AnsiString');
14792:  Function StringToWide( const Value : AnsiString) : WideString');
14793:  Function WideToString( const Value : WideString) : AnsiString');
14794: end;
14795:
14796: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14797: begin
14798:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14799:  Procedure WakeOnLan( MAC, IP : string)');
14800:  Function GetDNS : string');
14801:  Function GetIEProxy( protocol : string) : TProxySetting');
14802:  Function GetLocalIPs : string');
14803: end;
14804:
14805:
14806: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14807: begin
14808:  AddConstantN('synCR','Char #$0d);
14809:  Const('synLF','Char #$0a);
14810:  Const('cSerialChunk','LongInt'( 8192);
14811:  Const('LockfileDirectory','String '/var/lock');
14812:  Const('PortIsClosed','LongInt'( - 1);
14813:  Const('ErrAlreadyOwned','LongInt'( 9991);
14814:  Const('ErrAlreadyInUse','LongInt'( 9992);
14815:  Const('ErrWrongParameter','LongInt'( 9993);
14816:  Const('ErrPortNotOpen','LongInt'( 9994);
14817:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14818:  Const('ErrMaxBuffer','LongInt'( 9996);
14819:  Const('ErrTimeout','LongInt'( 9997);
14820:  Const('ErrNotRead','LongInt'( 9998);
14821:  Const('ErrFrame','LongInt'( 9999);
14822:  Const('ErrOverrun','LongInt'( 10000);
14823:  Const('ErrRxOver','LongInt'( 10001);
14824:  Const('ErrRxParity','LongInt'( 10002);
14825:  Const('ErrTxFull','LongInt'( 10003);
14826:  Const('dcb_Binary','LongWord')( $00000001);
14827:  Const('dcb_ParityCheck','LongWord')( $00000002);
14828:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14829:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14830:  Const('dcb_DtrControlMask','LongWord')( $00000030);
```

```
14831:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14832:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14833:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14834:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14835:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14836:  Const('dcb_OutX','LongWord')( $00000100);
14837:  Const('dcb_InX','LongWord')( $00000200);
14838:  Const('dcb_ErrorChar','LongWord')( $00000400);
14839:  Const('dcb_NullStrip','LongWord')( $00000800);
14840:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14841:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14842:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14843:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14844:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14845:  Const('dcb_AbortOnError','LongWord')( $00004000);
14846:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14847:  Const('synSB1','LongInt'( 0);
14848:  Const('SB1andHalf','LongInt'( 1);
14849:  Const('synSB2','LongInt'( 2);
14850:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14851:  Const('CS7fix','LongWord')( $0000020);
14852:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14853:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14854:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14855:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14856:   //AddTypeS('PDCB', '^TDCB // will not work');
14857:  //Const('MaxRates','LongInt'( 18);
14858:  //Const('MaxRates','LongInt'( 30);
14859:  //Const('MaxRates','LongInt'( 19);
14860:  Const('O_SYNC','LongWord')( $0080);
14861:  Const('synOK','LongInt'( 0);
14862:  Const('synErr','LongInt'( integer ( - 1 ));
14863:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14864:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14865:   SIRegister_ESynaSerError(CL);
14866:   SIRegister_TBlockSerial(CL);
14867:  Function GetSerialPortNames : string');
14868: end;
14869:
14870: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14871: begin
14872:  Const('DLLIconvName','String 'libiconv.so');
14873:  Const('DLLIconvName','String 'iconv.dll');
14874:   AddTypeS('size_t', 'Cardinal');
14875:   AddTypeS('iconv_t', 'Integer');
14876:   //AddTypeS('iconv_t', 'Pointer');
14877:   AddTypeS('argptr', 'iconv_t');
14878:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14879:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14880:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14881:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14882:  Function SynaIconvClose( var cd : iconv_t) : integer');
14883:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14884:  Function IsIconvloaded : Boolean');
14885:  Function InitIconvInterface : Boolean');
14886:  Function DestroyIconvInterface : Boolean');
14887:  Const('ICONV_TRIVIALP','LongInt'( 0);
14888:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14889:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14890:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14891:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14892: end;
14893:
14894: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14895: begin
14896:  Const 'ICMP_ECHO','LongInt'( 8);
14897:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14898:  Const('ICMP_UNREACH','LongInt'( 3);
14899:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14900:  Const('ICMP6_ECHO','LongInt'( 128);
14901:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14902:  Const('ICMP6_UNREACH','LongInt'( 1);
14903:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14904:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14905:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14906:   SIRegister_TPINGSend(CL);
14907:  Function PingHost( const Host : string) : Integer');
14908:  Function TraceRouteHost( const Host : string) : string');
14909: end;
14910:
14911: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14912: begin
14913:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14914:  Const('synASN1_INT','LongWord')( $02);
14915:  Const('synASN1_OCTSTR','LongWord')( $04);
14916:  Const('synASN1_NULL','LongWord')( $05);
14917:  Const('synASN1_OBJID','LongWord')( $06);
14918:  Const('synASN1_ENUM','LongWord')( $0a);
```

```
14919:  Const('synASN1_SEQ','LongWord')( $30);
14920:  Const('synASN1_SETOF','LongWord')( $31);
14921:  Const('synASN1_IPADDR','LongWord')( $40);
14922:  Const('synASN1_COUNTER','LongWord')( $41);
14923:  Const('synASN1_GAUGE','LongWord')( $42);
14924:  Const('synASN1_TIMETICKS','LongWord')( $43);
14925:  Const('synASN1_OPAQUE','LongWord')( $44);
14926:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14927:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14928:  Function synASNEncLen( Len : Integer) : AnsiString');
14929:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14930:  Function synASNEncInt( Value : Integer) : AnsiString');
14931:  Function synASNEncUInt( Value : Integer) : AnsiString');
14932:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14933:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14934:  Function synMibToId( Mib : String) : AnsiString');
14935:  Function synIdToMib( const Id : AnsiString) : String');
14936:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14937:  Function ASNdump( const Value : AnsiString) : AnsiString');
14938:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14939:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14940: end;
14941:
14942: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14943: begin
14944:  Const('cLDAPProtocol','String '389');
14945:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14946:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14947:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14948:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14949:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14950:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14951:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14952:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14953:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14954:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14955:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14956:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14957:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14958:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14959:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14960:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14961:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14962:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14963:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14964:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14965:   SIRegister_TLDAPAttribute(CL);
14966:   SIRegister_TLDAPAttributeList(CL);
14967:   SIRegister_TLDAPResult(CL);
14968:   SIRegister_TLDAPResultList(CL);
14969:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14970:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14971:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14972:   SIRegister_TLDAPSend(CL);
14973:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14974: end;
14975:
14976:
14977: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14978: begin
14979:  Const('cSysLogProtocol','String '514');
14980:  Const('FCL_Kernel','LongInt'( 0);
14981:  Const('FCL_UserLevel','LongInt'( 1);
14982:  Const('FCL_MailSystem','LongInt'( 2);
14983:  Const('FCL_System','LongInt'( 3);
14984:  Const('FCL_Security','LongInt'( 4);
14985:  Const('FCL_Syslogd','LongInt'( 5);
14986:  Const('FCL_Printer','LongInt'( 6);
14987:  Const('FCL_News','LongInt'( 7);
14988:  Const('FCL_UUCP','LongInt'( 8);
14989:  Const('FCL_Clock','LongInt'( 9);
14990:  Const('FCL_Authorization','LongInt'( 10);
14991:  Const('FCL_FTP','LongInt'( 11);
14992:  Const('FCL_NTP','LongInt'( 12);
14993:  Const('FCL_LogAudit','LongInt'( 13);
14994:  Const('FCL_LogAlert','LongInt'( 14);
14995:  Const('FCL_Time','LongInt'( 15);
14996:  Const('FCL_Local0','LongInt'( 16);
14997:  Const('FCL_Local1','LongInt'( 17);
14998:  Const('FCL_Local2','LongInt'( 18);
14999:  Const('FCL_Local3','LongInt'( 19);
15000:  Const('FCL_Local4','LongInt'( 20);
15001:  Const('FCL_Local5','LongInt'( 21);
15002:  Const('FCL_Local6','LongInt'( 22);
15003:  Const('FCL_Local7','LongInt'( 23);
15004:   Type(TSyslogSeverity','(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15005:   SIRegister_TSyslogMessage(CL);
15006:   SIRegister_TSyslogSend(CL);
15007:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
      Content:string):Boolean;
```

```
15008: end;
15009:
15010:
15011: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
15012: begin
15013:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15014:   SIRegister_TMessHeader(CL);
15015:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
15016:   SIRegister_TMimeMess(CL);
15017: end;
15018:
15019: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
15020: begin
15021:   (FindClass('TOBJECT'),'TMimePart');
15022:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15023:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15024:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15025:   SIRegister_TMimePart(CL);
15026:  Const('MaxMimeType','LongInt'( 25);
15027:  Function GenerateBoundary : string');
15028: end;
15029:
15030: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15031: begin
15032:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
15033:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
15034:  Function NeedInline( const Value : AnsiString) : boolean');
15035:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15036:  Function InlineCode( const Value : string) : string');
15037:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15038:  Function InlineEmail( const Value : string) : string');
15039: end;
15040:
15041: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15042: begin
15043:  Const('cFtpProtocol','String '21');
15044:  Const('cFtpDataProtocol','String '20');
15045:  Const('FTP_OK','LongInt'( 255);
15046:  Const('FTP_ERR','LongInt'( 254);
15047:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15048:   SIRegister_TFTPListRec(CL);
15049:   SIRegister_TFTPList(CL);
15050:   SIRegister_TFTPSend(CL);
15051:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15052:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15053:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
       ToPort, ToFile, ToUser, ToPass : string) : Boolean');
15054: end;
15055:
15056: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15057: begin
15058:  Const('cHttpProtocol','String '80');
15059:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15060:   SIRegister_THTTPSend(CL);
15061:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
15062:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
15063:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
15064:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
15065:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
       ResultData:TStrings):Bool;
15066: end;
15067:
15068: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15069: begin
15070:  Const('cSmtpProtocol','String '25');
15071:   SIRegister_TSMTPSend(CL);
15072:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
       Passw:string):Bool;
15073:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15074:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
       Username, Password : string):Boolean');
15075: end;
15076:
15077: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15078: begin
15079:  Const('cSnmpProtocol','String '161');
15080:  Const('cSnmpTrapProtocol','String '162');
15081:  Const('SNMP_V1','LongInt'( 0);
15082:  Const('SNMP_V2C','LongInt'( 1);
15083:  Const('SNMP_V3','LongInt'( 3);
15084:  Const('PDUGetRequest','LongWord')( $A0);
15085:  Const('PDUGetNextRequest','LongWord')( $A1);
15086:  Const('PDUGetResponse','LongWord')( $A2);
15087:  Const('PDUSetRequest','LongWord')( $A3);
15088:  Const('PDUTrap','LongWord')( $A4);
15089:  Const('PDUGetBulkRequest','LongWord')( $A5);
15090:  Const('PDUInformRequest','LongWord')( $A6);
15091:  Const('PDUTrapV2','LongWord')( $A7);
15092:  Const('PDUReport','LongWord')( $A8);
```

```
15093:  Const('ENoError',LongInt 0;
15094:  Const('ETooBig','LongInt')( 1);
15095:  Const('ENoSuchName','LongInt'( 2);
15096:  Const('EBadValue','LongInt'( 3);
15097:  Const('EReadOnly','LongInt'( 4);
15098:  Const('EGenErr','LongInt'( 5);
15099:  Const('ENoAccess','LongInt'( 6);
15100:  Const('EWrongType','LongInt'( 7);
15101:  Const('EWrongLength','LongInt'( 8);
15102:  Const('EWrongEncoding','LongInt'( 9);
15103:  Const('EWrongValue','LongInt'( 10);
15104:  Const('ENoCreation','LongInt'( 11);
15105:  Const('EInconsistentValue','LongInt'( 12);
15106:  Const('EResourceUnavailable','LongInt'( 13);
15107:  Const('ECommitFailed','LongInt'( 14);
15108:  Const('EUndoFailed','LongInt'( 15);
15109:  Const('EAuthorizationError','LongInt'( 16);
15110:  Const('ENotWritable','LongInt'( 17);
15111:  Const('EInconsistentName','LongInt'( 18);
15112:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15113:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15114:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15115:   SIRegister_TSNMPMib(CL);
15116:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15117:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15118:   SIRegister_TSNMPRec(CL);
15119:   SIRegister_TSNMPSend(CL);
15120:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15121:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15122:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15123:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15124:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15125:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15126:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : TStringList) : Integer');
15127: end;
15128:
15129: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15130: begin
15131:  Function GetDomainName2: AnsiString');
15132:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15133:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15134:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15135:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15136:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15137: end;
15138:
15139: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15140: begin
15141:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15142:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15143:  Function wwStrToDate( const S : string) : boolean');
15144:  Function wwStrToTime( const S : string) : boolean');
15145:  Function wwStrToDateTime( const S : string) : boolean');
15146:  Function wwStrToTimeVal( const S : string) : TDateTime');
15147:  Function wwStrToDateVal( const S : string) : TDateTime');
15148:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15149:  Function wwStrToInt( const S : string) : boolean');
15150:  Function wwStrToFloat( const S : string) : boolean');
15151:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15152:  Function wwNextDay( Year, Month, Day : Word) : integer');
15153:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15154:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15155:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15156:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15157:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15158:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15159:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15160:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15161:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15162: end;
15163:
15164: unit uPSI_Themes;
15165: Function ThemeServices : TThemeServices');
15166: Function ThemeControl( AControl : TControl) : Boolean');
15167: Function UnthemedDesigner( AControl : TControl) : Boolean');
15168: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15169: begin
15170:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15171: end;
15172: Unit uPSC_menus;
15173:  Function StripHotkey( const Text : string) : string');
15174:  Function GetHotkey( const Text : string) : string');
15175:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15176:  Function IsAltGRPressed : boolean');
15177:
15178: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15179: begin
```

```
15180:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15181:   SIRegister_TIdIMAP4Server(CL);
15182: end;
15183:
15184: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15185: begin
15186:  'HASH_SIZE','LongInt'( 256);
15187:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15188:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15189:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15190:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15191:    +' : Integer; Value : Variant; end');
15192:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15193:   SIRegister_TVariantSymbolTable(CL);
15194: end;
15195:
15196: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15197: begin
15198:   SIRegister_TThreadLocalVariables(CL);
15199:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15200:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15201:  Function ThreadLocals : TThreadLocalVariables');
15202:  Procedure WriteDebug( sz : String)');
15203:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15204:  'UDF_FAILURE','LongInt'( 1);
15205:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15206:  CL.AddTypeS('mTByteArray', 'array of byte;');
15207:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15208:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15209:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15210:  function IsNetworkConnected: Boolean;
15211:  function IsInternetConnected: Boolean;
15212:  function IsCOMConnected: Boolean;
15213:  function IsNetworkOn: Boolean;
15214:  function IsInternetOn: Boolean;
15215:  function IsCOMOn: Boolean;
15216:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15217:  TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15218:  Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT');
15219:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15220:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15221:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15222:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15223:  Function GetMenu( hWnd : HWND) : HMENU');
15224:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15225: end;
15226:
15227: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15228: begin
15229:   SIRegister_IDataBlock(CL);
15230:   SIRegister_ISendDataBlock(CL);
15231:   SIRegister_ITransport(CL);
15232:   SIRegister_TDataBlock(CL);
15233:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15234:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15235:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15236:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15237:   SIRegister_TCustomDataBlockInterpreter(CL);
15238:   SIRegister_TSendDataBlock(CL);
15239:   'CallSig','LongWord')( $D800);
15240:   'ResultSig','LongWord')( $D400);
15241:   'asMask','LongWord')( $00FF);
15242:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15243:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15244:  Procedure CheckSignature( Sig : Integer)');
15245: end;
15246:
15247: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15248: begin
15249:   //CL.AddTypeS('HINTERNET', '___Pointer');
15250:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15251:   CL.AddTypeS('HINTERNET', 'Integer');
15252:   CL.AddTypeS('HINTERNET2', '___Pointer');
15253:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15254:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15255:   CL.AddTypeS('INTERNET_PORT', 'Word');
15256:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15257:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15258:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15259:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15260:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15261:  Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
      lpUrlComponents:TURLComponents):BOOL;
15262:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
      dwUrlLength:DWORD):BOOL;
15263:  Function InternetCloseHandle(hInet : HINTERNET) : BOOL');
15264:  Function
      InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
      lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
```

```
15265:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15266:  ;dwContext:DWORD):HINTERNET;
15267:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
        lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15268:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
        dwContext:DWORD):BOOL;
15269:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15270:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15271:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15272:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15273:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15274:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15275:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15276:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
        lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15277:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
        lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15278:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15279:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15280:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
        dwContext:DWORD):BOOL;
15281:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
        TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15282:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
        BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15283:  Function
        WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15284:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15285:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15286:  Function
        FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15287:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15288:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15289:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15290:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15291:  Function
        FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15292:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15293:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15294:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15295:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15296:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15297:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15298:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15299:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15300:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15301:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15302:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15303:  Function
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
        PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15304:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15305:  Function
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15306:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
        PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15307:  Function
        HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15308:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
        dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15309:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15310:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15311:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15312:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15313:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15314:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15315:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15316:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15317:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
        lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15318:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15319:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15320:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15321:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15322: end;
15323:
15324: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15325: begin
15326:   AddTypeS('strCharSet', 'set of char');
15327:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15328:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15329:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15330:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15331:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15332:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15333:  Procedure strStripWhiteSpace( var s : string)');
```

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

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

```
15510:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15511:  Function String_ToInt( Str : String) : Integer');
15512:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15513:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15514:  Function String_Reverse( Str : String) : String');
15515:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15516:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15517:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15518:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15519:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15520:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15521:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15522:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15523:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15524:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15525:  Function String_End( Str : String; Cars : Word) : String');
15526:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15527:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15528:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15529:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15530:  Function String_IsNumbers( Str : String) : Boolean');
15531:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15532:  Function StringToCsvCell( aStr : String) : String');
15533: end;
15534:
15535: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15536: begin
15537:  Function LongDayName( aDate : TDate) : String');
15538:  Function LongMonthName( aDate : TDate) : String');
15539:  Function ShortYearOf( aDate : TDate) : byte');
15540:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15541:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15542:  Function SecondsToMinutes( Seconds : Integer) : Double');
15543:  Function MinutesToSeconds( Minutes : Double) : Integer');
15544:  Function MinutesToHours( Minutes : Integer) : Double');
15545:  Function HoursToMinutes( Hours : Double) : Integer');
15546:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15547:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15548:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15549:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15550:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15551:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15552:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15553:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15554:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15555: end;
15556:
15557: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15558: begin
15559:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15560:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15561:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15562:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15563:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15564:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15565:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15566:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15567:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15568:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15569:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15570:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15571:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15572:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15573:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15574:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15575:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15576:  Procedure cyCenterControl( aControl : TControl)');
15577:  Function GetLastParent( aControl : TControl) : TWinControl');
15578:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15579:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15580: end;
15581:
15582: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15583: begin
15584:  Function TablePackTable( Tab : TTable) : Boolean');
15585:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15586:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15587:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15588:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15589:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15590:  Function TableEmptyTable( Tab : TTable) : Boolean');
15591:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15592:  Procedure TableFindNearest( aTable : TTable; Value : String)');
```

```
15593:  Function
        TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
15594:  Function
        TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15595:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15596: end;
15597:
15598: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15599: begin
15600:   SIRegister_TcyRunTimeDesign(CL);
15601:   SIRegister_TcyShadowText(CL);
15602:   SIRegister_TcyBgPicture(CL);
15603:   SIRegister_TcyGradient(CL);
15604:   SIRegister_tcyBevel(CL);
15605:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15606:   SIRegister_tcyBevels(CL);
15607:   SIRegister_TcyImagelistOptions(CL);
15608:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15609: end;
15610:
15611: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15612: begin
15613:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15614:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15615:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15616:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15617:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15618:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15619:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15620:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15621:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
       BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15622:  Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
       BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Bool;const DrawRight:Bool;const
       DrawBottom:Bool;const RoundRect:bool;
15623:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15624:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15625:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
       TColor; aState : TButtonState; Focused, Hot : Boolean)');
15626:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
       GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15627:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
       const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15628:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
       TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15629:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
       TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15630:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
       WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15631:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15632:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15633:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15634:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
       CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15635:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15636:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15637:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15638:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15639:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15640:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15641:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15642:  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);
15643:  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;
15644:  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);');
15645:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15646:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15647:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15648:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15649:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15650:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15651:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15652:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15653:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
```

```
15654:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15655:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15656:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15657:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15658:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15659:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15660:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15661: end;
15662:
15663: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15664: begin
15665:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15666:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15667:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15668:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15669:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15670:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15671:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15672:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15673:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15674:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15675:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15676:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
       bmInvertReverseFromColor);
15677:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
       rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15678:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15679:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15680: end;
15681:
15682: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15683: begin
15684:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15685:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15686:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15687:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15688:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15689:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15690:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15691:  Const SC_GROUP_IDENTIFIERW','String') '+');
15692:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15693:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15694:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15695:  Const SERVICE_INACTIVE','LongWord $00000002);
15696:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15697:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15698:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15699:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15700:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15701:  Const SERVICE_STOPPED','LongWord $00000001);
15702:  Const SERVICE_START_PENDING','LongWord $00000002);
15703:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15704:  Const SERVICE_RUNNING','LongWord $00000004);
15705:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15706:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15707:  Const SERVICE_PAUSED','LongWord $00000007);
15708:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15709:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15710:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15711:  Const SC_MANAGER_CONNECT','LongWord $0001);
15712:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15713:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15714:  Const SC_MANAGER_LOCK','LongWord $0008);
15715:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15716:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15717:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15718:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15719:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15720:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15721:  Const SERVICE_START','LongWord $0010);
15722:  Const SERVICE_STOP','LongWord $0020);
15723:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15724:  Const SERVICE_INTERROGATE','LongWord $0080);
15725:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15726:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15727:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15728:  Const SERVICE_ADAPTER','LongWord $00000004);
15729:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15730:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15731:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15732:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15733:  Const SERVICE_BOOT_START','LongWord $00000000);
15734:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15735:  Const SERVICE_AUTO_START','LongWord $00000002);
15736:  Const SERVICE_DEMAND_START','LongWord $00000003);
15737:  Const SERVICE_DISABLED','LongWord $00000004);
15738:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15739:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15740:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
```

```
15741:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15742:   CL.AddTypeS('SC_HANDLE', 'THandle');
15743:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15744:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15745:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15746:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15747:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15748:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15749:   Const TServiceStatus', '_SERVICE_STATUS');
15750:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15751:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15752:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15753:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15754:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15755:   TEnumServiceStatus', 'TEnumServiceStatusA');
15756:   SC_LOCK', '___Pointer');
15757:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15758:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15759:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15760:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15761:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15762:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15763:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15764:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15765:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15766:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15767:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15768:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15769:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15770:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15771:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15772:   TQueryServiceConfig', 'TQueryServiceConfigA');
15773:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15774:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15775:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15776:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15777:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15778:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15779:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15780:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
      TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15781:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
      lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
      : DWORD):BOOL');
15782:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
      lpcchBuffer:DWORD):BOOL');
15783:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
      lpcchBuffer:DWORD):BOOL;
15784:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15785:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15786:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15787:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15788:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
      cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15789:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15790:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15791:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15792:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15793: end;
15794:
15795: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15796: begin
15797:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
      AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
      BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15798:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
      DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15799:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15800:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
      TWinControl;
15801:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
      AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15802:  Function CreateNotifyThread(const
      FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15803: end;
15804:
15805: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15806: begin
15807:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15808:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15809:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15810:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15811:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15812:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15813:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15814:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15815:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
```

```
15816:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15817:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15818:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15819:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15820:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15821:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15822:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15823:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15824:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15825:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15826:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15827:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15828:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15829:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15830:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15831:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15832:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15833:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15834:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15835:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15836:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15837:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15838:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15839:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15840:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15841:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15842:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15843:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15844:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15845:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15846:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15847:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15848:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15849:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15850:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15851:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15852:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
      List:TStrings):Bool
15853:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15854:   FindClass('TOBJECT'),'EJclFileSummaryError');
15855:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15856:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15857:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15858:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15859:   SIRegister_TJclFilePropertySet(CL);
15860:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15861:   SIRegister_TJclFileSummary(CL);
15862:   SIRegister_TJclFileSummaryInformation(CL);
15863:   SIRegister_TJclDocSummaryInformation(CL);
15864:   SIRegister_TJclMediaFileSummaryInformation(CL);
15865:   SIRegister_TJclMSISummaryInformation(CL);
15866:   SIRegister_TJclShellSummaryInformation(CL);
15867:   SIRegister_TJclStorageSummaryInformation(CL);
15868:   SIRegister_TJclImageSummaryInformation(CL);
15869:   SIRegister_TJclDisplacedSummaryInformation(CL);
15870:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15871:   SIRegister_TJclMiscSummaryInformation(CL);
15872:   SIRegister_TJclWebViewSummaryInformation(CL);
15873:   SIRegister_TJclMusicSummaryInformation(CL);
15874:   SIRegister_TJclDRMSummaryInformation(CL);
15875:   SIRegister_TJclVideoSummaryInformation(CL);
15876:   SIRegister_TJclAudioSummaryInformation(CL);
15877:   SIRegister_TJclControlPanelSummaryInformation(CL);
15878:   SIRegister_TJclVolumeSummaryInformation(CL);
15879:   SIRegister_TJclShareSummaryInformation(CL);
15880:   SIRegister_TJclLinkSummaryInformation(CL);
15881:   SIRegister_TJclQuerySummaryInformation(CL);
15882:   SIRegister_TJclImageInformation(CL);
15883:   SIRegister_TJclJpegSummaryInformation(CL);
15884: end;
15885:
15886: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15887: begin
15888:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15889:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15890:   T8087Infinity','( icProjective, icAffine )');
15891:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15892:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15893:  Function Get8087ControlWord : Word');
15894:  Function Get8087Infinity : T8087Infinity');
15895:  Function Get8087Precision : T8087Precision');
15896:  Function Get8087Rounding : T8087Rounding');
15897:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15898:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15899:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15900:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15901:  Function Set8087ControlWord( const Control : Word) : Word');
15902:  Function ClearPending8087Exceptions : T8087Exceptions');
15903:  Function GetPending8087Exceptions : T8087Exceptions');
```

```
15904:  Function GetMasked8087Exceptions : T8087Exceptions');
15905:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15906:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15907:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15908: end;
15909:
15910: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15911: begin
15912:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15913:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15914:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
       Accept:Bool;Sorted:Bool;
15915:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15916:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15917:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15918:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15919:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15920: end;
15921:
15922: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15923: begin
15924:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15925:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15926:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15927:   type ULONG', 'Cardinal');
15928:      LPCWSTR', 'PChar');
15929:   CL.AddTypeS('LPWSTR', 'PChar');
15930:   LPSTR', 'PChar');
15931:   TBindVerb', 'ULONG');
15932:   TBindInfoF', 'ULONG');
15933:   TBindF', 'ULONG');
15934:   TBSCF', 'ULONG');
15935:   TBindStatus', 'ULONG');
15936:   TCIPStatus', 'ULONG');
15937:   TBindString', 'ULONG');
15938:   TPiFlags', 'ULONG');
15939:   TOIBdgFlags', 'ULONG');
15940:   TParseAction', 'ULONG');
15941:   TPSUAction', 'ULONG');
15942:   TQueryOption', 'ULONG');
15943:   TPUAF', 'ULONG');
15944:   TSZMFlags', 'ULONG');
15945:   TUrlZone', 'ULONG');
15946:   TUrlTemplate', 'ULONG');
15947:   TZAFlags', 'ULONG');
15948:   TUrlZoneReg', 'ULONG');
15949:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15950:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15951:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15952:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15953:  const 'CF_NULL','LongInt').SetInt( 0);
15954:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15955:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15956:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15957:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15958:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15959:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15960:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15961:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15962:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15963:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15964:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15965:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15966:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15967:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15968:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15969:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15970:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15971:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15972:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15973:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15974:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15975:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15976:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15977:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15978:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15979:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15980:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15981:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15982:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15983:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15984:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15985:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15986:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15987:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15988:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15989:   SIRegister_IPersistMoniker(CL);
15990:   SIRegister_IBindProtocol(CL);
15991:   SIRegister_IBinding(CL);
```

```
15992:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15993:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15994:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15995:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15996:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15997:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15998:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15999:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
16000:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
16001:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
16002:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
16003:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
16004:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16005:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16006:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16007:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
16008:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
16009:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
16010:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
16011:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
16012:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
16013:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
16014:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
16015:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
16016:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
16017:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
16018:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
16019:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16020:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16021:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16022:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16023:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
16024:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16025:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16026:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16027:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16028:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16029:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16030:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16031:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
16032:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
16033:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16034:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16035:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16036:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16037:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
16038:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
16039:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16040:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16041:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16042:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16043:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16044:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16045:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16046:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16047:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16048:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16049:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16050:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16051:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16052:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16053:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16054:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16055:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16056:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16057:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16058:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16059:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
16060:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16061:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16062:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16063:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16064:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16065:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16066:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16067:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16068:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16069:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16070:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
16071:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16072:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16073:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16074:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16075:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16076:  // PBindInfo', '^TBindInfo // will not work');
16077:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16078:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16079:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16080:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
```

```
16081:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16082:   TBindInfo', '_tagBINDINFO');
16083:   BINDINFO', '_tagBINDINFO');}
16084:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16085:    +'criptor : DWORD; bInheritHandle : BOOL; end');
16086:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16087:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16088:   //PRemBindInfo', '^TRemBindInfo // will not work');
16089:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16090:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16091:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16092:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16093:    +'n; dwReserved : DWORD; end');
16094:   TRemBindInfo', '_tagRemBINDINFO');
16095:   RemBINDINFO', '_tagRemBINDINFO');}
16096:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
16097:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16098:   TRemFormatEtc', 'tagRemFORMATETC');
16099:   RemFORMATETC', 'tagRemFORMATETC');
16100:   SIRegister_IBindStatusCallback(CL);
16101:   SIRegister_IAuthenticate(CL);
16102:   SIRegister_IHttpNegotiate(CL);
16103:   SIRegister_IWindowForBindingUI(CL);
16104:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16105:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16106:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16107:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16108:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16109:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16110:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16111:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16112:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16113:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16114:   SIRegister_ICodeInstall(CL);
16115:   SIRegister_IWinInetInfo(CL);
16116:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16117:   SIRegister_IHttpSecurity(CL);
16118:   SIRegister_IWinInetHttpInfo(CL);
16119:   SIRegister_IBindHost(CL);
16120:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16121:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16122:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16123:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16124:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16125:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
      : IBindStatusCallback) : HResult');
16126:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
     IBindStatusCallback) : HResult');
16127:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
     p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16128:  Function HlinkGoBack( unk : IUnknown) : HResult');
16129:  Function HlinkGoForward( unk : IUnknown) : HResult');
16130:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16131: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16132:   SIRegister_IInternet(CL);
16133:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16134:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16135:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16136:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16137:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16138:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16139:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16140:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16141:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16142:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16143:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16144:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16145:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16146:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16147:   //POLEStrArray', '^TOLESTRArray // will not work');
16148:   SIRegister_IInternetBindInfo(CL);
16149:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16150:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16151:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16152:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16153:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16154:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16155:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16156:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16157:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16158:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16159:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16160:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16161:   //PProtocolData', '^TProtocolData // will not work');
16162:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16163:   TProtocolData', '_tagPROTOCOLDATA');
16164:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16165:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16166:   SIRegister_IInternetProtocolRoot(CL);
```

```
16167:   SIRegister_IInternetProtocol(CL);
16168:   SIRegister_IInternetProtocolSink(CL);
16169:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16170:   SIRegister_IInternetSession(CL);
16171:   SIRegister_IInternetThreadSwitch(CL);
16172:   SIRegister_IInternetPriority(CL);
16173:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16174:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16175:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16176:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16177:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16178:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16179:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16180:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16181:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16182:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16183:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16184:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16185:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16186:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16187:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16188:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16189:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16190:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16191:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16192:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16193:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16194:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16195:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16196:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16197:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16198:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16199:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16200:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16201:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16202:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16203:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16204:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16205:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16206:   SIRegister_IInternetProtocolInfo(CL);
16207:   IOInet', 'IInternet');
16208:   IOInetBindInfo', 'IInternetBindInfo');
16209:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16210:   IOInetProtocol', 'IInternetProtocol');
16211:   IOInetProtocolSink', 'IInternetProtocolSink');
16212:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16213:   IOInetSession', 'IInternetSession');
16214:   IOInetPriority', 'IInternetPriority');
16215:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16216:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16217:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16218:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16219:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16220:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16221:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16222:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16223:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16224:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16225:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16226:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16227:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16228:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16229:    //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16230:   //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16231: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16232: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16233:   //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16234:   //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16235:   //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16236:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16237:   SIRegister_IInternetSecurityMgrSite(CL);
16238:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16239:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16240:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16241:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16242:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16243:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16244:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16245:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16246:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
```

```
16247:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16248:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16249:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16250:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16251:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16252:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16253:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16254:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16255:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16256:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16257:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16258:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16259:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16260:  const 'PUAF_NOUIFLOCKED','LongWord').SetUInt( $00100000);
16261:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16262:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16263:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16264:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16265:   SIRegister_IInternetSecurityManager(CL);
16266:   SIRegister_IInternetHostSecurityManager(CL);
16267:   SIRegister_IInternetSecurityManagerEx(CL);
16268:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16269:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16270:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16271:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16272:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16273:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16274:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16275:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16276:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16277:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16278:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16279:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16280:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16281:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16282:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16283:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16284:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16285:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16286:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16287:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16288:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16289:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16290:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16291:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16292:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16293:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16294:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16295:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16296:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16297:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16298:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16299:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16300:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16301:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16302:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16303:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16304:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16305:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16306:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16307:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16308:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16309:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16310:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16311:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16312:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16313:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16314:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16315:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16316:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16317:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16318:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16319:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16320:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16321:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16322:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16323:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16324:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16325:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16326:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16327:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16328:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16329:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16330:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16331:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16332:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16333:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16334:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16335:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
```

```
16336:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16337:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16338:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16339:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16340:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16341:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16342:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16343:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16344:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16345:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16346:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16347:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16348:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16349:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16350:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16351:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16352:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16353:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16354:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16355:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16356:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16357:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16358:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16359:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16360:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16361:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16362:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16363:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16364:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16365:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16366:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16367:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16368:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16369:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16370:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16371:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16372:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16373:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16374:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16375:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16376:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16377:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16378:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16379:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16380:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16381:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16382:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16383:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16384:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16385:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16386:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16387:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16388:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16389:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16390:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16391:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16392:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16393:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16394:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16395:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16396:   //PZoneAttributes', '^TZoneAttributes // will not work');
16397:  _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');
16398:   { _ZONEATTRIBUTES = packed record
16399:     cbSize: ULONG;
16400:     szDisplayName: array [0..260 - 1] of WideChar;
16401:     szDescription: array [0..200 - 1] of WideChar;
16402:     szIconPath: array [0..260 - 1] of WideChar;
16403:     dwTemplateMinLevel: DWORD;
16404:     dwTemplateRecommended: DWORD;
16405:     dwTemplateCurrentLevel: DWORD;
16406:     dwFlags: DWORD;
16407:   end; }
16408:   TZoneAttributes', '_ZONEATTRIBUTES');
16409:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16410:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16411:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16412:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16413:   SIRegister_IInternetZoneManager(CL);
16414:   SIRegister_IInternetZoneManagerEx(CL);
16415:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16416:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16417:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16418:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16419:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16420:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16421:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16422:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
```

```
16423:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16424:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16425:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16426:   TCodeBaseHold', '_tagCODEBASEHOLD');
16427:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16428:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16429:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16430:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16431:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16432:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16433:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16434:   TSoftDistInfo', '_tagSOFTDISTINFO');
16435:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16436:   SIRegister_ISoftDistExt(CL);
16437:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16438:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
      dwAdvertisedVersionLS : DWORD) : HResult');
16439:   SIRegister_IDataFilter(CL);
16440:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16441:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16442:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16443:    +'terFlags : DWORD; end');
16444:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16445:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16446:   //PDataInfo', '^TDataInfo // will not work');
16447:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16448:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16449:   TDataInfo', '_tagDATAINFO');
16450:   DATAINFO', '_tagDATAINFO');
16451:   SIRegister_IEncodingFilterFactory(CL);
16452:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16453:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16454:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16455:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16456:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16457:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16458:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16459:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16460:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16461: end;
16462:
16463: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16464: begin
16465:  Procedure reformatMemo( const m : TCustomMemo)');
16466:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16467:  Procedure MoveToTop( memo : TMemo)');
16468:  Procedure ScrollToTop( memo : TMemo)');
16469:  Function LineNumberClicked( memo : TMemo) : integer');
16470:  Function MemoClickedLine( memo : TMemo) : integer');
16471:  Function ClickedMemoLine( memo : TMemo) : integer');
16472:  Function MemoLineClicked( memo : TMemo) : integer');
16473:  Function LinePositionClicked( Memo : TMemo) : integer');
16474:  Function ClickedMemoPosition( memo : TMemo) : integer');
16475:  Function MemoPositionClicked( memo : TMemo) : integer');
16476:  Procedure AdjustGridSize( grid : TDrawGrid)');
16477:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16478:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16479:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16480:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16481:  Procedure sortstrDown( var s : string)');
16482:  Procedure sortstrUp( var s : string)');
16483:  Procedure rotatestrleft( var s : string)');
16484:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16485:  Function deblank( s : string) : string');
16486:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16487:  Procedure FreeAndClearListBox( C : TListBox);');
16488:  Procedure FreeAndClearMemo( C : TMemo);');
16489:  Procedure FreeAndClearStringList( C : TStringList);');
16490:  Function dffgetfilesize( f : TSearchrec) : int64');
16491: end;
16492:
16493: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16494: begin
16495:   CL.AddTypeS('intset', 'set of byte');
16496:   TPoint64', 'record x : int64; y : int64; end');
16497:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16498:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16499:  Function GeneratePentagon( n : integer) : integer');
16500:  Function IsPentagon( p : integer) : boolean');
16501:  Function isSquare( const N : int64) : boolean');
16502:  Function isCube( const N : int64) : boolean');
16503:  Function isPalindrome( const n : int64) : boolean;');
16504:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16505:  Function GetEulerPhi( n : int64) : int64');
16506:  Function dffIntPower( a, b : int64) : int64;');
16507:  Function IntPower1( a : extended; b : int64) : extended;');
16508:  Function gcd2( a, b : int64) : int64');
16509:  Function GCDMany( A : array of integer) : integer');
16510:  Function LCMMany( A : array of integer) : integer');
```

```
16511:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16512:  Function dffFactorial( n : int64) : int64');
16513:  Function digitcount( n : int64) : integer');
16514:  Function nextpermute( var a : array of integer) : boolean');
16515:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16516:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16517:  Function InttoBinaryStr( nn : integer) : string');
16518:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16519:  Function AngleToStr( angle : extended) : string');
16520:  Function deg2rad( deg : extended) : extended');
16521:  Function rad2deg( rad : extended) : extended');
16522:  Function GetLongToMercProjection( const long : extended) : extended');
16523:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16524:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16525:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16526:   SIRegister_TPrimes(CL);
16527:   //RIRegister_TPrimes(CL);
16528:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16529:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16530:  Function Random64( const N : Int64) : Int64;');
16531:  Procedure Randomize64');
16532:  Function Random641 : extended;');
16533:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16534: end;
16535:
16536: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16537: begin
16538:   TrealPoint', 'record x : extended; y : extended; end');
16539:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16540:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16541:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16542:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16543:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16544:  Function realpoint( x, y : extended) : TRealPoint');
16545:  Function dist( const p1, p2 : TrealPoint) : extended');
16546:  Function intdist( const p1, p2 : TPoint) : integer');
16547:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16548:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16549:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16550:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16551:  Function GetTheta( const L : TLine) : extended;');
16552:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16553:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16554:  Procedure Extendline( var L : TLine; dist : integer);');
16555:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16556:  Function Linesintersect( line1, line2 : TLine) : boolean');
16557:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16558:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16559:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16560:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16561:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16562:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16563:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16564:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16565:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16566:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16567:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16568:  Function DegtoRad( d : extended) : extended');
16569:  Function RadtoDeg( r : extended) : extended');
16570:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16571:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16572:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16573:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16574:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16575:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16576:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16577:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16578:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16579: end;
16580:
16581:
16582: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16583: begin
16584:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16585:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16586:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16587:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16588:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16589:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16590:    +'arth : extended; Phase : extended; end');
16591:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16592:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16593:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16594:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
```

```
16595:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16596:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16597:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16598:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16599:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16600:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16601:   SIRegister_TAstronomy(CL);
16602:  Function AngleToStr( angle : extended) : string');
16603:  Function StrToAngle( s : string; var angle : extended) : boolean');
16604:  Function HoursToStr24( t : extended) : string');
16605:  Function RPoint( x, y : extended) : TRPoint');
16606:  Function getStimename( t : TDTType) : string');
16607: end;
16608:
16609: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16610: begin
16611:   TCardValue', 'Integer');
16612:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16613:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16614:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16615:   SIRegister_TCard(CL);
16616:   SIRegister_TDeck(CL);
16617: end;
16618:
16619: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16620: begin
16621:   tMethodCall', 'Procedure');
16622:   tVerboseCall', 'Procedure ( s : string)');
16623:  // PTEdge', '^TEdge // will not work');
16624:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16625:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16626:   SIRegister_TNode(CL);
16627:   SIRegister_TGraphList(CL);
16628: end;
16629:
16630: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16631: begin
16632:   ParserFloat', 'extended');
16633:   //PParserFloat', '^ParserFloat // will not work');
16634:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16635:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16636:   //POperation', '^TOperation // will not work');
16637:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16638:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16639:    +'; Token : TDFFToken; end');
16640:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16641:   (CL.FindClass('TOBJECT'),'EMathParserError');
16642:   CL.FindClass('TOBJECT'),'ESyntaxError');
16643:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16644:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16645:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16646:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16647:   (CL.FindClass('TOBJECT'),'EBadName');
16648:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16649:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16650:   SIRegister_TCustomParser(CL);
16651:   SIRegister_TExParser(CL);
16652: end;
16653:
16654:   function isService: boolean;
16655:   begin
16656:     result:= NOT(Application is TApplication);
16657:     {result:= Application is TServiceApplication;}
16658:   end;
16659:   function isApplication: boolean;
16660:   begin
16661:     result:= Application is TApplication;
16662:   end;
16663:   //SM_REMOTESESSION = $1000
16664:   function isTerminalSession: boolean;
16665:   begin
16666:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16667:   end;
16668:
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 : Strin'
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:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
```

```
16682:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16683:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16684:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16685:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16686:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16687:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16688:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16689:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16690:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16691: end;
16692:
16693:
16694: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16695: begin
16696:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16697:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16698:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16699:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16700:    +'inationsRepeat, CombinationsRepeatDown )');
16701:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16702:   SIRegister_TComboSet(CL);
16703: end;
16704:
16705:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16706: begin
16707:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16708:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16709:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16710:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16711:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16712:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16713:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16714:   SIRegister_TcyBaseComm(CL);
16715:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16716:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16717:  Function ValidateFileMappingName( aName : String) : String');
16718:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16719: end;
16720:
16721: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16722: begin
16723:   CL.AddTypeS('DERString', 'String');
16724:   CL.AddTypeS('DERChar', 'Char');
16725:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16726:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16727:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16728:   CL.AddTypeS('DERNString', 'String');
16729:  const DERDecimalSeparator','String').SetString( '.');
16730:  const DERDefaultChars','String')('+°@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16731:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16732:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16733:  Function isValidWebMailChar( aChar : Char) : Boolean');
16734:  Function isValidwebSite( aStr : String) : Boolean');
16735:  Function isValidWebMail( aStr : String) : Boolean');
16736:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16737:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16738:  Function IsDERChar( aChar : Char) : Boolean');
16739:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16740:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16741:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16742:  Function IsDERSymbols( aDERString : String) : Boolean');
16743:  Function StringToDERCharSet( aStr : String) : DERString');
16744:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16745:  Function IsDERNChar( aChar : Char) : Boolean');
16746:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16747:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16748:  Function DERExtractWebMail( aDERStr : DERString) : String');
16749:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16750:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16751:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
      RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16752:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
      aElementsType : TElementsType) : String;');
16753:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
      Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16754: end;
16755:
16756: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16757: begin
16758:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16759:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16760:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16761:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16762:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16763:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16764:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
```

```
16765:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16766:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
        RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16767:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
        PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
        PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16768:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
        SingleDestinationColor, RefreshBmp : Boolean)');
16769:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16770:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16771:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16772:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16773:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16774:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16775:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16776:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16777: end;
16778:
16779: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16780: begin
16781:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16782:   TPCMChannel', '( cMono, cStereo )');
16783:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16784:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16785:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16786:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16787:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16788:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16789:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16790:    +'it48000Hz, Stereo16bit48000Hz )');
16791:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16792:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16793:    tWaveFormatEx', 'PWaveFormatEx');
16794:    HMMIO', 'Integer');
16795:   TWaveDeviceFormats', 'set of TPCMFormat');
16796:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16797:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16798:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16799:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16800:   TWaveOutOptions', 'set of TWaveOutOption');
16801:   TStreamOwnership2', '( soReference, soOwned )');
16802:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16803:  // PRawWave', '^TRawWave // will not work');
16804:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16805:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16806:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
16807:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16808:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16809:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16810:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16811:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16812:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16813:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16814:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16815:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16816:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16817:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16818:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16819:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16820:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16821:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16822:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16823:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16824:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16825:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16826:  GetAudioFormat( FormatTag : Word) : String');
16827:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16828:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16829:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16830:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16831:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16832:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16833:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16834:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
        TObject; BufferSize : DWORD) : Boolean');
16835:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
        dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16836:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
        TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16837:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16838:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16839:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16840:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16841:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16842:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16843: end;
16844:
16845: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16846: begin
```

```
16847:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16848:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16849:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16850:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
16851:  'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16852:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16853:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16854:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16855:   CL.AddTypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg )');
16856:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16857:   SIRegister_TNamedPipe(CL);
16858:   SIRegister_TServerPipe(CL);
16859:   SIRegister_TClientPipe(CL);
16860:  CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD) : DWORD');
16861:  Function HasOverlappedIoCompleted( const ov : OVERLAPPED) : Boolean');
16862:  Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean):
      TOverlappedResult;
16863:  Function GetStreamAsText( stm : TStream) : string');
16864:  Procedure SetStreamAsText( const aTxt : string; stm : TStream)');
16865: end;
16866:
16867: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16868: begin
16869:  // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16870:   SIRegister_TThumbData(CL);
16871:  'PIC_BMP','LongInt').SetInt( 0);
16872:  'PIC_JPG','LongInt').SetInt( 1);
16873:  'THUMB_WIDTH','LongInt').SetInt( 60);
16874:  'THUMB_HEIGHT','LongInt').SetInt( 60);
16875:  Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean');
16876:  Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)');
16877:  Function ReadBitmap( Filehandle : Integer; Width, Height : Integer) : TBitmap');
16878:  Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap)');
16879:  Function OpenPicture( fn : string; var tp : Integer) : Integer');
16880:  Function ConvertPicture( pi : Integer; tp : Integer) : TBitmap');
16881:  Function LoadPicture( fn : string; var w, h : Integer) : TBitmap');
16882:  Function TurnBitmap( bmp : TBitmap; ang : Integer) : TBitmap');
16883:  Function RotateBitmap( Bitmap : TBitmap; Direction : Integer) : TBitmap');
16884:  Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect');
16885:  Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer) : TBitmap');
16886:  Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer)');
16887:  Procedure FindFiles( path, mask : string; items : TStringList)');
16888:  Function LetFileName( s : string) : string');
16889:  Function LetParentPath( path : string) : string');
16890:  Function AddBackSlash( path : string) : string');
16891:  Function CutBackSlash( path : string) : string');
16892: end;
16893:
16894: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16895: begin
16896:  //'BYTES','LongInt').SetInt( 1);
16897:  'KBYTES','LongInt').SetInt( 1024);
16898:  'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ));
16899:  'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ));
16900:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16901:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
16902:   SIRegister_MakeComServerMethodsPublic(CL);
16903:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16904:  Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) : Boolean');
16905:  Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)');
16906:  Function TBGetTempFolder : string');
16907:  Function TBGetTempFile : string');
16908:  Function TBGetModuleFilename : string');
16909:  Function FormatModuleVersionInfo( const aFilename : string) : string');
16910:  Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD) : string');
16911:  Function TBGetFileSize( aFile : string; aMultipleOf : Integer) : Integer');
16912:  Function FormatAttribString( aAttr : Integer) : string');
16913:  Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo) : string');
16914:  Function ShellRecycle( aWnd : HWND; aFileOrFolder : string) : Boolean');
16915:  Function IsDebuggerPresent : BOOL');
16916:  Function TBNotImplemented : HRESULT');
16917: end;
16918:
16919: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16920: begin
16921:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16922:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16923:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16924:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16925:   //TDrivesProperty = array['A'..'Z'] of boolean;
16926:  Function TBSetSystemTime( DateTime : TDateTime; DOW : word) : boolean');
16927:  Function IsElevated : Boolean');
16928:  Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16929:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16930:  Function TrimNetResource( UNC : string) : string');
16931:  Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16932:  Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16933:  Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
      UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
```

```
16934:  Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16935:  Function TBIsWindowsVista : Boolean');
16936:  Procedure SetVistaFonts( const AForm : TForm)');
16937:  Procedure SetVistaContentFonts( const AFont : TFont)');
16938:  Function GetProductType( var sType : String) : Boolean');
16939:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16940:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16941:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16942:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16943:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16944:  Function lstrlen( lpString : PChar) : Integer');
16945:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16946:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16947: end;
16948:
16949: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16950: begin
16951:  'cLineTerminator','Char').SetString( #10);
16952:  'cLineTerminators','String').SetString( #13#10);
16953:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16954:   SIRegister_TFixedCriticalSection(CL);
16955:   SIRegister_TMultiReadSingleWrite(CL);
16956:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16957:  Function GetDecimalSeparator : Char');
16958:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16959:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16960:   CL.AddTypeS('NativeInt', 'Integer');
16961:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16962:   CL.AddTypeS('NativeUInt', 'Cardinal');
16963:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16964:   //CL.AddTypeS('TBytes', 'array of Byte');
16965:   CL.AddTypeS('RawByteString', 'UnicodeString');
16966:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16967:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16968:   SIRegister_TPath(CL);
16969:   SIRegister_TFile(CL);
16970:   SIRegister_TdwsThread(CL);
16971:  Function GetSystemMilliseconds : Int64');
16972:  Function UTCDateTime : TDateTime');
16973:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16974:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16975:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16976:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16977:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16978:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16979:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16980:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16981:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16982:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16983:  Function InterlockedIncrement( var val : Integer) : Integer');
16984:  Function InterlockedDecrement( var val : Integer) : Integer');
16985:  Procedure FastInterlockedIncrement( var val : Integer)');
16986:  Procedure FastInterlockedDecrement( var val : Integer)');
16987:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16988:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16989:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16990:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16991:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16992:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16993:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16994:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16995:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16996:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16997:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16998:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16999:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
17000:  Procedure CloseFileHandle( hFile : THandle)');
17001:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
17002:  Function FileMove( const existing, new : UnicodeString) : Boolean');
17003:  Function dwsFileDelete( const fileName : String) : Boolean');
17004:  Function FileRename( const oldName, newName : String) : Boolean');
17005:  Function dwsFileSize( const name : String) : Int64');
17006:  Function dwsFileDateTime( const name : String) : TDateTime');
17007:  Function DirectSet8087CW( newValue : Word) : Word');
17008:  Function DirectSetMXCSR( newValue : Word) : Word');
17009:  Function TtoObject( const T: byte) : TObject');
17010:  Function TtoPointer( const T: byte) : ___Pointer');
17011:  Procedure GetMemForT(var T: byte; Size : integer)');
17012:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
17013: end;
17014:
17015: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
17016: begin
17017:  'IPStrSize','LongInt').SetInt( 15);
17018:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17019:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
```

```
17020:  'ADWSBASE','LongInt').SetInt( 9000);
17021:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17022:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17023:   SIRegister_EApdSocketException(CL);
17024:   TWsMode', '( wsClient, wsServer )');
17025:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket');
17026:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer)');
17027:   SIRegister_TApdSocket(CL);
17028: end;
17029:
17030: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
17031: begin
17032:   SIRegister_TApdCustomComPort(CL);
17033:   SIRegister_TApdComPort(CL);
17034:  Function ComName( const ComNumber : Word) : ShortString');
17035:  Function SearchComPort( const C : TComponent) : TApdCustomComPort');
17036: end;
17037:
17038: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
17039: begin
17040:  Function inAddBackslash( const S : String) : String');
17041:  Function PathChangeExt( const Filename, Extension : String) : String');
17042:  Function PathCharCompare( const S1, S2 : PChar) : Boolean');
17043:  Function PathCharIsSlash( const C : Char) : Boolean');
17044:  Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean');
17045:  Function PathCharLength( const S : String; const Index : Integer) : Integer');
17046:  Function inPathCombine( const Dir, Filename : String) : String');
17047:  Function PathCompare( const S1, S2 : String) : Integer');
17048:  Function PathDrivePartLength( const Filename : String) : Integer');
17049:  Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
17050:  Function inPathExpand( const Filename : String) : String');
17051:  Function PathExtensionPos( const Filename : String) : Integer');
17052:  Function PathExtractDir( const Filename : String) : String');
17053:  Function PathExtractDrive( const Filename : String) : String');
17054:  Function PathExtractExt( const Filename : String) : String');
17055:  Function PathExtractName( const Filename : String) : String');
17056:  Function PathExtractPath( const Filename : String) : String');
17057:  Function PathIsRooted( const Filename : String) : Boolean');
17058:  Function PathLastChar( const S : String) : PChar');
17059:  Function PathLastDelimiter( const Delimiters, S : string) : Integer');
17060:  Function PathLowercase( const S : String) : String');
17061:  Function PathNormalizeSlashes( const S : String) : String');
17062:  Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17063:  Function PathPos( Ch : Char; const S : String) : Integer');
17064:  Function PathStartsWith( const S, AStartsWith : String) : Boolean');
17065:  Function PathStrNextChar( const S : PChar) : PChar');
17066:  Function PathStrPrevChar( const Start, Current : PChar) : PChar');
17067:  Function PathStrScan( const S : PChar; const C : Char) : PChar');
17068:  Function inRemoveBackslash( const S : String) : String');
17069:  Function RemoveBackslashUnlessRoot( const S : String) : String');
17070:  Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean)');
17071: end;
17072:
17073:
17074: procedure SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17075: begin
17076:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17077:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17078:  NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17079:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17080:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName');
17081:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString');
17082:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17083:   //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17084:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17085:   SIRegister_TOneShotTimer(CL);
17086:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17087:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17088:  Function NewFileExists( const Name : String) : Boolean');
17089:  Function inDirExists( const Name : String) : Boolean');
17090:  Function FileOrDirExists( const Name : String) : Boolean');
17091:  Function IsDirectoryAndNotReparsePoint( const Name : String) : Boolean');
17092:  Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String');
17093:  Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17094:  Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filename:String): Boolean');
17095:  Function IniKeyExists( const Section, Key, Filename : String) : Boolean');
17096:  Function IsIniSectionEmpty( const Section, Filename : String) : Boolean');
17097:  Function SetIniString( const Section, Key, Value, Filename : String) : Boolean');
17098:  Function SetIniInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean');
17099:  Function SetIniBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean');
17100:  Procedure DeleteIniEntry( const Section, Key, Filename : String)');
17101:  Procedure DeleteIniSection( const Section, Filename : String)');
17102:  Function GetEnv( const EnvVar : String) : String');
17103:  Function GetCmdTail : String');
17104:  Function GetCmdTailEx( StartIndex : Integer) : String');
17105:  Function NewParamCount : Integer');
17106:  Function NewParamStr( Index : Integer) : string');
17107:  Function AddQuotes( const S : String) : String');
17108:  Function RemoveQuotes( const S : String) : String');
```

```
17109:  Function inGetShortName( const LongName : String) : String');
17110:  Function inGetWinDir : String');
17111:  Function inGetSystemDir : String');
17112:  Function GetSysWow64Dir : String');
17113:  Function GetSysNativeDir( const IsWin64 : Boolean) : String');
17114:  Function inGetTempDir : String');
17115:  Function StringChange( var S : String; const FromStr, ToStr : String) : Integer');
17116:  Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17117:  Function AdjustLength( var S : String; const Res : Cardinal) : Boolean');
17118:  Function UsingWinNT : Boolean');
17119:  Function ConvertConstPercentStr( var S : String) : Boolean');
17120:  Function ConvertPercentStr( var S : String) : Boolean');
17121:  Function ConstPos( const Ch : Char; const S : String) : Integer');
17122:  Function SkipPastConst( const S : String; const Start : Integer) : Integer');
17123:  Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String) : Boolean');
17124:  Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String):Boolean;
17125:  Function RegValueExists( H : HKEY; Name : PChar) : Boolean');
17126:  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;
17127:  Function RegOpenKeyExView(const
       RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17128:  Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar) : Longint;
17129:  Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17130:  Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17131:  Function GetShellFolderPath( const FolderID : Integer) : String');
17132:  Function IsAdminLoggedOn : Boolean');
17133:  Function IsPowerUserLoggedOn : Boolean');
17134:  Function IsMultiByteString( const S : AnsiString) : Boolean');
17135:  Function FontExists( const FaceName : String) : Boolean');
17136:  //Procedure FreeAndNil( var Obj)');
17137:  Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE');
17138:  Function GetUILanguage : LANGID');
17139:  Function RemoveAccelChar( const S : String) : String');
17140:  Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17141:  Function AddPeriod( const S : String) : String');
17142:  Function GetExceptMessage : String');
17143:  Function GetPreferredUIFont : String');
17144:  Function IsWildcard( const Pattern : String) : Boolean');
17145:  Function WildcardMatch( const Text, Pattern : PChar) : Boolean');
17146:  Function IntMax( const A, B : Integer) : Integer');
17147:  Function Win32ErrorString( ErrorCode : Integer) : String');
17148:  Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17149:  Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean');
17150:  Function DeleteDirTree( const Dir : String) : Boolean');
17151:  Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17152:  Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17153:  // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17154:  Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean');
17155:  Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean');
17156:  Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean');
17157:  Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean');
17158:  Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17159:  Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean');
17160:  Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)');
17161: end;
17162:
17163: procedure SIRegister_CmnFunc(CL: TPSPascalCompiler);
17164: begin
17165:   SIRegister_TWindowDisabler(CL);
17166:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17167:   TMsgBoxCallbackFunc','procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17168:  Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17169:  Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17170:  Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer');
17171:  Function MsgBoxP( const Text,Caption: PChar; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17172:  Function inMsgBox(const Text,Caption:String; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17173:  Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
       Typ:TMsgBoxType;const Buttons:Cardinal):Integer');
17174:  Procedure ReactivateTopWindow');
17175:  Procedure SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar)');
17176:  Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)');
17177:  Procedure SetMessageBoxCallbackFunc(const AFunc : TMsgBoxCallbackFunc; const AParam : LongInt)');
17178: end;
17179:
17180: procedure SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17181: begin
17182:   SIRegister_TImageGrabber(CL);
17183:   SIRegister_TCaptureDrivers(CL);
17184:   SIRegister_TCaptureDriver(CL);
17185: end;
17186:
17187: procedure SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17188: begin
17189:  Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
       Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean');
17190:  Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
       Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean');
17191: end;
```

```
17192:
17193: procedure SIRegister_RedirFunc(CL: TPSPascalCompiler);
17194: begin
17195:   CL.AddTypeS('TPreviousFsRedirectionState', record DidDisable : Boolean; OldValue : ___Pointer; end');
17196:  CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17197:  Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Boo;
17198:  Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17199:  Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17200:  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;
17201:  Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
       FailIfExists : BOOL) : BOOL');
17202:  Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17203:  Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17204:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17205:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
       TWin32FindData) : THandle');
17206:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17207:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17208:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
       TFileVersionNumbers) : Boolean');
17209:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17210:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17211:  Function MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilen,NewFilename:String;const
       Flags:DWORD):BOOL;
17212:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17213:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17214:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17215:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17216:   SIRegister_TFileRedir(CL);
17217:   SIRegister_TTextFileReaderRedir(CL);
17218:   SIRegister_TTextFileWriterRedir(CL);
17219: end;
17220:
17221: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17222: begin
17223:   //CL.AddTypeS('LongWord', 'Cardinal');
17224:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17225:  Function Compare64( const N1, N2 : Integer64) : Integer');
17226:  Procedure Dec64( var X : Integer64; N : LongWord)');
17227:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17228:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17229:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
17230:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17231:  Function Integer64ToStr( X : Integer64) : String');
17232:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17233:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17234:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17235:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17236:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17237: end;
17238:
17239: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17240: begin
17241:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17242:   SIRegister_TSimpleStringList(CL);
17243:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17244:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17245:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17246:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17247:  // TMD5Digest = array[0..15] of Byte;
17248:   //     TSHA1Digest = array[0..19] of Byte;
17249:  Function CheckForMutexes( Mutexes : String) : Boolean');
17250:  Function CreateTempDir : String');
17251:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17252:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
       FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17253:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
       DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
       TDeleteFileProc; const Param : Pointer) : Boolean');
17254:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
       TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
       TDetermineDefaultLanguageResult');
17255:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17256:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17257:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17258:  Function GetComputerNameString : String');
17259:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17260:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17261:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17262: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17263:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17264:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17265: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17266:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17267:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
       TotBytes:Int64):Bool;
```

```
17268:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
       TotalBytes: Integer64):Bool;
17269:  Function GetUserNameString : String');
17270:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17271:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
       String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
       ResultCode:Integer) : Boolean');
17272: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
       TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17273:  Procedure InternalError( const Id : String)');
17274:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17275:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17276:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17277:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17278:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17279:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
17280:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17281:  Procedure RefreshEnvironment');
17282:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17283:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17284:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17285:  Function RestartComputer : Boolean');
17286:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17287:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17288:  Procedure Win32ErrorMsg( const FunctionName : String)');
17289:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17290:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17291:  //from Func2
17292:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
       IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
17293:   //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17294:  Procedure RegisterTypeLibrary( const Filename : String)');
17295:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17296:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17297:  function getVersionInfoEx3: TOSVersionInfoEx;');
17298:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17299:  procedure InitOle;');
17300:  Function ExpandConst( const S : String) : String');
17301:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17302:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17303:  Function ExpandConstIfPrefixed( const S : String) : String');
17304:  Procedure LogWindowsVersion');
17305:  Function EvalCheck( const Expression : String) : Boolean');
17306: end;
17307:
17308: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17309: begin
17310:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17311:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17312:   SIRegister_TResourceModule(CL);
17313:   SIRegister_TResourceDetails(CL);
17314:   SIRegister_TAnsiResourceDetails(CL);
17315:   SIRegister_TUnicodeResourceDetails(CL);
17316:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17317:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17318:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17319:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17320:  Function ResourceNameToInt( const s : string) : Integer');
17321:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17322: end;
17323:
17324:
17325: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17326: begin
17327:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17328:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
17329:     RegisterMethod('Constructor Create');
17330:     RegisterMethod('Procedure Free');
17331:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17332:     RegisterMethod('Procedure WriteString( const S : String)');
17333:     RegisterMethod('Procedure ReadString( var S : String)');
17334:   end;
17335:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17336:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17337:       SimpleComPort.WriteString(AsciiChar);
17338: end;
17339:
17340:
17341: procedure SIRegister_Console(CL: TPSPascalCompiler);
17342: begin
17343:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17344: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17345:  ('conBW40','LongInt').SetInt( 0);
17346:  ('conCO40','LongInt').SetInt( 1);
17347:  ('conBW80','LongInt').SetInt( 2);
17348:  ('conCO80','LongInt').SetInt( 3);
17349:  ('conMono','LongInt').SetInt( 7);
17350:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17351:  //CL.AddConstantN('C40','').SetString( CO40);
```

```
17352:  //CL.AddConstantN('C80','').SetString( C080);
17353:  Function conReadKey : Char');
17354:  Function conKeyPressed : Boolean');
17355:  Procedure conGotoXY( X, Y : Smallint)');
17356:  Function conWhereX : Integer');
17357:  Function conWhereY : Integer');
17358:  Procedure conTextColor( Color : Byte);');
17359:  Function conTextColor1 : Byte;');
17360:  Procedure conTextBackground( Color : Byte);');
17361:  Function conTextBackground1 : Byte;');
17362:  Procedure conTextMode( Mode : Word)');
17363:  Procedure conLowVideo');
17364:  Procedure conHighVideo');
17365:  Procedure conNormVideo');
17366:  Procedure conClrScr');
17367:  Procedure conClrEol');
17368:  Procedure conInsLine');
17369:  Procedure conDelLine');
17370:  Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17371:  Function conScreenWidth : Smallint');
17372:  Function conScreenHeight : Smallint');
17373:  Function conBufferWidth : Smallint');
17374:  Function conBufferHeight : Smallint');
17375:  procedure InitScreenMode;');
17376: end;
17377:
17378: (*----------------------------------------------------------------------------*)
17379: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17380: begin
17381:   SIRegister_TNoRefCountObject(CL);
17382:  Procedure FreeObjects( List : TFPList)');
17383:  Procedure GetMethodList( AObject : TObject; AList : TStrings);');
17384:  Procedure GetMethodList1( AClass : TClass; AList : TStrings);');
17385: end;
17386:
17387: procedure SIRegister_ToolsUnit(CL: TPSPascalCompiler);
17388: begin
17389:  'MaxDataSet','LongInt').SetInt( 35);
17390:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17391:   SIRegister_TDBConnector(CL);
17392:   SIRegister_TDBBasicsTestSetup(CL);
17393:   SIRegister_TTestDataLink(CL);
17394:  'testValuesCount','LongInt').SetInt( 25);
17395:  Procedure InitialiseDBConnector');
17396:  Procedure FreeDBConnector');
17397:  Function DateTimeToTimeString( d : tdatetime) : string');
17398:  Function TimeStringToDateTime( d : String) : TDateTime');
17399: end;
17400:
17401: procedure SIRegister_fpcunit(CL: TPSPascalCompiler);
17402: begin
17403:   SIRegister_EAssertionFailedError(CL);
17404:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17405:   CL.AddTypeS('TRunMethod', 'Procedure');
17406:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTestResult');
17407:   SIRegister_TTest(CL);
17408:   SIRegister_TAssert(CL);
17409:   SIRegister_TTestFailure(CL);
17410:   SIRegister_ITestListener(CL);
17411:   SIRegister_TTestCase(CL);
17412:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17413:   SIRegister_TTestSuite(CL);
17414:   SIRegister_TTestResult(CL);
17415:  Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17416: end;
17417:
17418: procedure SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
17419: begin
17420:   TOBJECT'),'ETCPBuffer');
17421:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Intege'
17422:    +r; Head : Integer; Used : Integer; end');
17423:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17424:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17425:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
17426:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
17427:  Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int;
17428:  Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17429:  Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17430:  Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17431:  Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17432:  Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17433:  Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17434:  Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17435:  Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17436:  Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17437:  Function TCPBufferRemove(var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17438:  Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17439:  Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17440:  Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');
```

```
17441:  Function TCPBufferEmpty( const TCPBuf : TTCPBuffer) : Boolean');
17442:  Function TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Integer');
17443:  Function TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer');
17444:  Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer)');
17445: end;
17446:
17447: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17448: begin
17449:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17450:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17451:  CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
17452:  CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
17453:  CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0);
17454:  CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
17455:  CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
17456:  Procedure LoadGlut( const dll : String)');
17457:  Procedure FreeGlut');
17458: end;
17459:
17460: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17461: begin
17462:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17463:  Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean) : THandle');
17464: end;
17465:
17466: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17467: begin
17468:   TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )');
17469:   TTLedState', '( LedOn, LedOff, LedDisabled )');
17470:   SIRegister_TSwitchLed(CL);
17471:  //CL.AddDelphiFunction('Procedure Register');
17472: end;
17473:
17474: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17475: begin
17476:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17477:    +'xisting, fdOpenAlways, fdTruncateExisting )');
17478:   CL.AddTypeS('TIFileAccess', '( faRead, faWrite, faReadWrite )');
17479:   CL.AddTypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17480:   SIRegister_TCustomFile(CL);
17481:   SIRegister_TIFile(CL);
17482:   SIRegister_TMemoryFile(CL);
17483:   SIRegister_TTextFileReader(CL);
17484:   SIRegister_TTextFileWriter(CL);
17485:   SIRegister_TFileMapping(CL);
17486:   SIRegister_EFileError(CL);
17487: end;
17488:
17489: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17490: begin
17491:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17492:    +', ffaDirectory, ffaArchive, ffaAnyFile )');
17493:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17494:   SIRegister_TFileSearch(CL);
17495: end;
17496:
17497: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17498: begin
17499:  TRGBType', 'record RedHex : string; GreenHex : string; BlueHex :'
17500:    + string; Red : integer; Green : integer; Blue : integer; end');
17501:  Function FadeColor( aColor : Longint; aFade : integer) : Tcolor');
17502:  Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17503: end;
17504:
17505: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17506: begin
17507:  Procedure SaveOscSettings');
17508:  Procedure GetOscSettings');
17509: end;
17510:
17511: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17512: begin
17513:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer)');
17514:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17515:    FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17516:    64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17517:    Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17518:   SIRegister_TcyDebug(CL);
17519: end;
17520:
17521: (*----------------------------------------------------------------------------*)
17522: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17523: begin
17524:   TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )');
17525:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )');
17526:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )');
17527:   SIRegister_TDestinationOptions(CL);
17528:   TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult)');
17529:   TProcOnCopyFileProgressEvent','Procedure(Sender:TObject;FileBytes,
       TransferedBytes:int64;PercentDone:Int64);
```

```
17530:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17531:   SIRegister_TcyCopyFiles(CL);
17532:  Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
       ResetAttr:boolean): TCopyFileResult');
17533:  Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
       TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17534:  Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
        ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;'
17535:                       + FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17536: end;
17537:
17538: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17539: begin
17540:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17541:   SIRegister_TcyFileAttributes(CL);
17542:   SIRegister_TSearchRecInstance(CL);
17543:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17544:   TOptions', 'set of TOption');
17545:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17546:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
       Accept:bool;
17547:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17548:   SIRegister_TcyCustomSearchFiles(CL);
17549:   SIRegister_TcySearchFiles(CL);
17550:  Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17551:  Function IscyFolder( aSRec : TSearchrec) : Boolean');
17552: end;
17553:
17554: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17555: begin
17556:  Function jCountChar( const s : string; ch : char) : integer');
17557:  Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings)');
17558:  Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17559:  Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17560:  Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17561:  Function jNormalizeDateSeparator( const s : string) : string');
17562:  Function jIsValidDateString( const Value : string) : boolean');
17563:  Function jIsValidTimeString( const Value : string) : boolean');
17564:  Function jIsValidDateTimeString( const Value : string) : boolean');
17565: end;
17566:
17567: procedure SIRegister_kcMapViewer(CL: TPSPascalCompiler);
17568: begin
17569:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapViewer');
17570:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
17571:    +'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
17572:    +', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
17573:    +'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
17574:    +'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17575:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17576:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17577:   TIntPoint', 'record X : Int64; Y : Int64; end');
17578:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17579:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17580:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17581:   SIRegister_TCustomDownloadEngine(CL);
17582:   SIRegister_TCustomGeolocationEngine(CL);
17583:   SIRegister_TMapViewer(CL);
17584: end;
17585:
17586: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17587: begin
17588:  (*CL.AddDelphiFunction('Function isFunc( name : TNamePart) : Boolean');*)
17589:  CL.AddDelphiFunction('Function isUnnamedFunc( name : TNamepart) : Boolean');
17590:  Function isPtrToFunc( name : TNamePart) : Boolean');
17591:  Function isFuncRetFuncPtr( name : TNamePart) : Boolean');
17592:  Function isPtrToFuncRetFuncPtr( name : TNamePart) : Boolean');
17593:  Function GetFuncParam( name : TNamePart) : TNamePart');
17594:  Function isArray( name : TNamePart) : Boolean');
17595:  Function GetArrayPart( name : TNamePart) : TNamePart');
17596:  Function GetIdFromPart( name : TNamePart) : AnsiString');
17597:  Function GetIdPart( name : TNamePart) : TNamePart');
17598:  Function isNamePartPtrToFunc( part : TNamePart) : Boolean');
17599:  Function isAnyBlock( part : TNamePart) : Boolean');*)
17600:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17601:   SIRegister_TLineBreaker(CL);
17602:   CL.AddTypeS('TNameKind', 'Integer');
17603:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17604:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17605:  Function SphericalMod( X : Extended) : Extended');
17606:  Function cSign( Value : Extended) : Extended');
17607:  Function LimitFloat( const eValue, eMin, eMax : Extended) : Extended');
17608:  Function AngleToRadians( iAngle : Extended) : Extended');
17609:  Function RadiansToAngle( eRad : Extended) : Extended');
17610:  Function Cross180( iLong : Double) : Boolean');
17611:  Function Mod180( Value : integer) : Integer');
17612:  Function Mod180Float( Value : Extended) : Extended');
17613:  Function MulDivFloat( a, b, d : Extended) : Extended');
17614:  Function LongDiff( iLong1, iLong2 : Double) : Double');
```

```
17615:  Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
17616:  Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17617:  Function FixFilePath( const Inpath, CheckPath : string) : string');
17618:  Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17619:  Procedure FillStringList( sl : TStringList; const aText : string)');
17620: end;
17621:
17622: procedure SIRegister_LedNumber(CL: TPSPascalCompiler);
17623: begin
17624:   TLedSegmentSize', 'Integer');
17625:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
17626:   SIRegister_TCustomLEDNumber(CL);
17627:   SIRegister_TLEDNumber(CL);
17628: end;
17629:
17630: procedure SIRegister_StStrL(CL: TPSPascalCompiler);
17631: begin
17632:   CL.AddTypeS('LStrRec', 'record AllocSize : Longint; RefCount : Longint; Length : Longint; end');
17633:    CL.AddTypeS('AnsiChar', 'Char');
17634:     CL.AddTypeS('BTable', 'array[0..255] of Byte');       //!!!
17635:   CL.AddDelphiFunction('Function HexBL( B : Byte) : AnsiString');
17636:  Function HexWL( W : Word) : AnsiString');
17637:  Function HexLL( L : LongInt) : AnsiString');
17638:  Function HexPtrL( P : ___Pointer) : AnsiString');
17639:  Function BinaryBL( B : Byte) : AnsiString');
17640:  Function BinaryWL( W : Word) : AnsiString');
17641:  Function BinaryLL( L : LongInt) : AnsiString');
17642:  Function OctalBL( B : Byte) : AnsiString');
17643:  Function OctalWL( W : Word) : AnsiString');
17644:  Function OctalLL( L : LongInt) : AnsiString');
17645:  Function Str2Int16L( const S : AnsiString; var I : SmallInt) : Boolean');
17646:  Function Str2WordL( const S : AnsiString; var I : Word) : Boolean');
17647:  Function Str2LongL( const S : AnsiString; var I : LongInt) : Boolean');
17648:  Function Str2RealL( const S : AnsiString; var R : Double) : Boolean');
17649:  Function Str2RealL( const S : AnsiString; var R : Real) : Boolean');
17650:  Function Str2ExtL( const S : AnsiString; var R : Extended) : Boolean');
17651:  Function Long2StrL( L : LongInt) : AnsiString');
17652:  Function Real2StrL( R : Double; Width : Byte; Places : ShortInt) : AnsiString');
17653:  Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : AnsiString');
17654:  Function ValPrepL( const S : AnsiString) : AnsiString');
17655:  Function CharStrL( C : Char; Len : Cardinal) : AnsiString');
17656:  Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17657:  Function PadLL( const S : AnsiString; Len : Cardinal) : AnsiString');
17658:  Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17659:  Function LeftPadL( const S : AnsiString; Len : Cardinal) : AnsiString');
17660:  Function TrimLeadL( const S : AnsiString) : AnsiString');
17661:  Function TrimTrailL( const S : AnsiString) : AnsiString');
17662:  Function TrimL( const S : AnsiString) : AnsiString');
17663:  Function TrimSpacesL( const S : AnsiString) : AnsiString');
17664:  Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17665:  Function CenterL( const S : AnsiString; Len : Cardinal) : AnsiString');
17666:  Function EntabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17667:  Function DetabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17668:  Function ScrambleL( const S, Key : AnsiString) : AnsiString');
17669:  Function SubstituteL( const S, FromStr, ToStr : AnsiString) : AnsiString');
17670:  Function FilterL( const S, Filters : AnsiString) : AnsiString');
17671:  Function CharExistsL( const S : AnsiString; C : AnsiChar) : Boolean');
17672:  Function CharCountL( const S : AnsiString; C : AnsiChar) : Cardinal');
17673:  Function WordCountL( const S, WordDelims : AnsiString) : Cardinal');
17674:  Function WordPositionL( N : Cardinal; const S, WordDelims : AnsiString; var Pos : Cardinal) : Boolean');
17675:  Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString) : AnsiString');
17676:  Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar) : Cardinal');
17677:  Function AsciiPositionL(N: Cardinal;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17678:  Function ExtractAsciiL( N : Cardinal; const S, WordDelims : AnsiString; Quote : AnsiChar) : AnsiString');
17679:  Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17680:  Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17681:  Function CompStringL( const S1, S2 : AnsiString) : Integer');
17682:  Function CompUCStringL( const S1, S2 : AnsiString) : Integer');
17683:  Function SoundexL( const S : AnsiString) : AnsiString');
17684:  Function MakeLetterSetL( const S : AnsiString) : Longint');
17685:  Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable)');
17686:  Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
       Pos:Card):Bool;
17687:  Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
       var Pos : Cardinal) : Boolean');
17688:  Function DefaultExtensionL( const Name, Ext : AnsiString) : AnsiString');
17689:  Function ForceExtensionL( const Name, Ext : AnsiString) : AnsiString');
17690:  Function JustFilenameL( const PathName : AnsiString) : AnsiString');
17691:  Function JustNameL( const PathName : AnsiString) : AnsiString');
17692:  Function JustExtensionL( const Name : AnsiString) : AnsiString');
17693:  Function JustPathnameL( const PathName : AnsiString) : AnsiString');
17694:  Function AddBackSlashL( const DirName : AnsiString) : AnsiString');
17695:  Function CleanPathNameL( const PathName : AnsiString) : AnsiString');
17696:  Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal) : Boolean');
17697:  Function CommaizeL( L : LongInt) : AnsiString');
17698:  Function CommaizeChL( L : Longint; Ch : AnsiChar) : AnsiString');
17699:  Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr,RtCurr:AnsiString;Sep,
       DecPt:Char):AnsiString;
17700:  Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
       RtCurr:AnsiString;Sep:Char):AnsiString);
```

```
17701:  Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal) : Boolean');
17702:  Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal) : Boolean');
17703:  Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17704:  Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal) : AnsiString');
17705:  Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal) : AnsiString');
17706:  Function StrChDeleteL( const S : AnsiString; Pos : Cardinal) : AnsiString');
17707:  Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17708:  Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17709:  Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17710:  Function CopyLeftL( const S : AnsiString; Len : Cardinal) : AnsiString');
17711:  Function CopyMidL( const S : AnsiString; First, Len : Cardinal) : AnsiString');
17712:  Function CopyRightL( const S : AnsiString; First : Cardinal) : AnsiString');
17713:  Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal) : AnsiString');
17714:  Function CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
       SubString:AString):Bool;
17715:  Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
       SubString:AnsiString):Bool;
17716:  Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean) : AnsiString');
17717:  Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
       var SubString : AnsiString) : Boolean');
17718:  Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
       SubString : AnsiString) : Boolean');
17719:  Function DeleteWithinL( const S, Delimiter : AnsiString) : AnsiString');
17720:  Function ExtractTokensL(const S,
       Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17721:  Function IsChAlphaL( C : AnsiChar) : Boolean');
17722:  Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17723:  Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17724:  Function IsStrAlphaL( const S : AnsiString) : Boolean');
17725:  Function IsStrNumericL( const S, Numbers : AnsiString) : Boolean');
17726:  Function IsStrAlphaNumericL( const S, Numbers : AnsiString) : Boolean');
17727:  Function KeepCharsL( const S, Chars : AnsiString) : AnsiString');
17728:  Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal) : Boolean');
17729:  Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal) : Boolean');
17730:  Function LastStringL( const S, AString : AnsiString; var Position : Cardinal) : Boolean');
17731:  Function LeftTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17732:  Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Card;var
       Replacements:Card):AnsiString;
17733:  Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
       Replacements:Cardinal):AnsiString');
17734:  Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
       Replacements:Cardinal):AnsiString;
17735:  Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17736:  Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17737:  Function RightTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17738:  Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal) : boolean');
17739:  Function TrimCharsL( const S, Chars : AnsiString) : AnsiString');
17740:  Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17741:  Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17742:  end;
17743:
17744:  procedure SIRegister_pwnative_out(CL: TPSPascalCompiler);
17745: begin
17746:  CL.AddConstantN('STDIN','LongInt').SetInt( 0);
17747:  ('STDOUT','LongInt').SetInt( 1);
17748:  ('STDERR','LongInt').SetInt( 2);
17749:  Procedure NativeWrite( s : astr);');
17750:  Procedure NativeWrite1( PString : PChar);');
17751:  Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal);');
17752:  Procedure NativeWriteLn( s : astr);');
17753:  Procedure NativeWriteLn1;');
17754: end;
17755:
17756: procedure SIRegister_synwrap1(CL: TPSPascalCompiler);
17757: begin
17758:   CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse
17759:    : integer; UltimateURL : ansistring; Headers : ansistring; end');
17760:   CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17761:  Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean): TSynwInfo;');
17762:  Function GetHttpFile1( const Url, UserAgent, Outfile : string) : TSynwInfo;');
17763:  Function GetHttpFile2( const Url, Outfile : string) : TSynwInfo;');
17764:  Function GetHttpFile3( const Url, outfile : string; verbose : boolean) : TSynwInfo;');
17765:  Function GetHtm( const Url : string) : string;');
17766:  Function GetHtm1( const Url, UserAgent : string) : string;');
17767:  Function GetUrl( const Url : string; verbose : boolean) : TSynwInfo;');
17768:  Function GetUrl1( const Url, useragent : string) : TSynwInfo;');
17769:  Function GetUrl2( const Url : string) : TSynwInfo;');
17770:  Function GetUrl3( const Url : string; const http : THTTPSend; verbose : boolean): TUrlInfo;');
17771:  Function GetUrl4( const Url : string; const http : THTTPSend) : TUrlInfo;');
17772:  Procedure StrToStream( s : String; strm : TMemoryStream)');
17773:  Function StrLoadStream( strm : TStream) : String');
17774:
17775: end;
17776:
17777: procedure SIRegister_HTMLUtil(CL: TPSPascalCompiler);
17778: begin
17779:  Function GetVal( const tag, attribname_ci : string) : string');
17780:  Function GetTagName( const Tag : string) : string');
17781:  Function GetUpTagName( const tag : string) : string');
```

```
17782:  Function GetNameValPair( const tag, attribname_ci : string) : string');
17783:  Function GetValFromNameVal( const namevalpair : string) : string');
17784:  Function GetNameValPair_cs( const tag, attribname : string) : string');
17785:  Function GetVal_JAMES( const tag, attribname_ci : string) : string');
17786:  Function GetNameValPair_JAMES( const tag, attribname_ci : string) : string');
17787:  Function CopyBuffer( StartIndex : PChar; Len : integer) : string');
17788:  Function Ucase( s : string) : string');
17789: end;
17790:
17791: procedure SIRegister_pwmain(CL: TPSPascalCompiler);
17792: begin
17793:  CL.AddConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT');
17794:  EXPIRED_COOKIE','String').SetString( 'Mon, 01 Jan 2001 12:00:00 GMT');
17795:  'SECURE_OFF','LongInt').SetInt( 0);
17796:  'SECURE_ON','LongInt').SetInt( 2);
17797:  'SECURE_FILTER','LongInt').SetInt( 3);
17798:   THandle or DWord!
17799:  //  astr = ansistring;
17800:   CL.AddTypeS('pastr', 'ansistring');
17801:   CL.AddTypeS('TFilterFunc', 'function(const s: pastr): pastr;');
17802:   uses pwinit at begin
17803:   //type  TFilterFunc = function(const s: astr): astr;
17804:  Demo: ..\maxbox3\examples2\519_powtils.txt
17805:
17806:  //CL.AddConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false);
17807:  //CL.AddConstantN('CASE_IGNORE','Boolean')BoolToStr( false);
17808:  Procedure pwInit');
17809:  Procedure OffReadln');
17810:  Function Lcase( const s : pastr) : pastr');
17811:  Function Ucase( const s : pastr) : pastr');
17812:  Function CountPostVars : longword');
17813:  Function GetPostVar( const name : pastr) : pastr;');
17814:  Function GetPostVar1( const name : pastr; filter : TFilterFunc) : pastr;');
17815:  Function GetPostVar_S( const name : pastr; Security : integer) : pastr');
17816:  Function GetPostVar_SF( const name : pastr; Security : integer) : pastr');
17817:  Function GetPostVarAsFloat( const name : pastr) : double');
17818:  Function GetPostVarAsInt( const name : pastr) : longint');
17819:  Function GetPostVar_SafeHTML( const name : pastr) : pastr');
17820:  Function FetchPostVarName( idx : longword) : pastr');
17821:  Function FetchPostVarVal( idx : longword) : pastr;');
17822:  Function FetchPostVarVal1( idx : longword; filter : TFilterFunc) : pastr;');
17823:  Function FetchPostVarName_S( idx : longword; Security : integer) : pastr');
17824:  Function FetchPostVarVal_S( idx : longword; Security : integer) : pastr');
17825:  Function IsPostVar( const name : pastr) : boolean');
17826:  Function CountAny : longword');
17827:  Function GetAny( const name : pastr) : pastr;');
17828:  Function GetAny1( const name : pastr; filter : TFilterFunc) : pastr;');
17829:  Function GetAny_S( const name : pastr; Security : integer) : pastr');
17830:  Function GetAnyAsFloat( const name : pastr) : double');
17831:  Function GetAnyAsInt( const name : pastr) : longint');
17832:  Function IsAny( const name : pastr) : byte');
17833:  Function CountCookies : longword');
17834:  Function FetchCookieName( idx : longword) : pastr');
17835:  Function FetchCookieVal( idx : longword) : pastr;');
17836:  Function FetchCookieVal1( idx : longword; filter : TFilterFunc) : pastr;');
17837:  Function GetCookie( const name : pastr) : pastr;');
17838:  Function GetCookie1( const name : pastr; filter : TFilterFunc) : pastr;');
17839:  Function GetCookieAsFloat( const name : pastr) : double');
17840:  Function GetCookieAsInt( const name : pastr) : longint');
17841:  Function IsCookie( const name : pastr) : boolean');
17842:  Function SetCookie( const name, value : pastr) : boolean');
17843:  Function SetCookieAsFloat( const name : pastr; value : double) : boolean');
17844:  Function SetCookieAsInt( const name : pastr; value : longint) : boolean');
17845:  Function SetCookieEx( const name, value, path, domain, expiry : pastr) : boolean');
17846:  Function SetCookieAsFloatEx(const name:pastr;value : double; const path,domain,expiry:pastr):bool;
17847:  Function SetCookieAsIntEx( const name:pastr;value : longint; const path,domain,expiry:pastr):bool;
17848:  Function UnsetCookie( const name : pastr) : boolean');
17849:  Function UnsetCookieEx( const name, path, domain : pastr) : boolean');
17850:  Function FilterHtml( const input : pastr) : pastr');
17851:  Function FilterHtml_S( const input : pastr; security : integer) : pastr');
17852:  Function TrimBadChars( const input : pastr) : pastr');
17853:  Function TrimBadFile( const input : pastr) : pastr');
17854:  Function TrimBadDir( const input : pastr) : pastr');
17855:  Function TrimBad_S( const input : pastr; security : integer) : pastr');
17856:  Function CountHeaders : longword');
17857:  Function FetchHeaderName( idx : longword) : pastr');
17858:  Function FetchHeaderVal( idx : longword) : pastr');
17859:  Function GetHeader( const name : pastr) : pastr');
17860:  Function IsHeader( const name : pastr) : boolean');
17861:  Function SetHeader( const name, value : pastr) : boolean');
17862:  Function UnsetHeader( const name : pastr) : boolean');
17863:  Function PutHeader( const header : pastr) : boolean');
17864:  Procedure Out1( const s : pastr)');
17865:  Procedure OutLn( const s : pastr)');
17866:  Procedure OutA( args : array of const)');
17867:  Procedure OutF( const s : pastr)');
17868:  Procedure OutLnF( const s : pastr)');
17869:  Procedure OutFF( const s : pastr)');
17870:  Procedure OutF_FI( const s : pastr; HTMLFilter : boolean)');
```

```
17871:  Procedure OutLnFF( const s : pastr)');
17872:  Procedure OutLnF_FI( const s : pastr; HTMLFilter : boolean)');
17873:  Function FileOut( const fname : pastr) : word');
17874:  Function ResourceOut( const fname : pastr) : word');
17875:  Procedure BufferOut( const buff, len : LongWord)');
17876:  Function TemplateOut( const fname : pastr; HtmlFilter : boolean) : word;');
17877:  Function TemplateOut1( const fname : pastr) : word;');
17878:  Function TemplateOut2( const fname : pastr; filter : TFilterFunc) : word;');
17879:  Function TemplateOut3( const fname : pastr; HtmlFilter : boolean) : word;');
17880:  Function TemplateRaw( const fname : pastr) : word');
17881:  Function Fmt( const s : pastr) : pastr;');
17882:  Function Fmt1( const s : pastr; filter : TFilterFunc) : pastr;');
17883:  Function FmtFilter( const s : pastr) : pastr');
17884:  Function Fmt_SF(const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecurity,TrimSecurity:int):pastr;
17885:  Function Fmt_SF1(const s:pastr; HTMLFilter:boolean; FilterSecurity, TrimSecurity : integer) : pastr;');
17886:  Function CountRtiVars : longword');
17887:  Function FetchRtiName( idx : longword) : pastr');
17888:  Function FetchRtiVal( idx : longword) : pastr');
17889:  Function GetRti( const name : pastr) : pastr');
17890:  Function GetRtiAsFloat( const name : pastr) : double');
17891:  Function GetRtiAsInt( const name : pastr) : longint');
17892:  Function IsRti( const name : pastr) : boolean');
17893:  Procedure SetRTI( const name, value : pastr)');
17894:  Function FetchUpfileName( idx : longword) : pastr');
17895:  Function GetUpfileName( const name : pastr) : pastr');
17896:  Function GetUpfileSize( const name : pastr) : longint');
17897:  Function GetUpfileType( const name : pastr) : pastr');
17898:  Function CountUpfiles : longword');
17899:  Function IsUpfile( const name : pastr) : boolean');
17900:  Function SaveUpfile( const name, fname : pastr) : boolean');
17901:  Function CountVars : longword');
17902:  Function FetchVarName( idx : longword) : pastr');
17903:  Function FetchVarVal( idx : longword) : pastr;');
17904:  Function FetchVarVal1( idx : longword; filter : TFilterFunc) : pastr;');
17905:  Function GetVar( const name : pastr) : pastr;');
17906:  Function GetVar1( const name : pastr; filter : TFilterFunc) : pastr;');
17907:  Function GetVar_S( const name : pastr; security : integer) : pastr');
17908:  Function GetVarAsFloat( const name : pastr) : double');
17909:  Function GetVarAsInt( const name : pastr) : longint');
17910:  Procedure SetVar( const name, value : pastr)');
17911:  Procedure SetVarAsFloat( const name : pastr; value : double)');
17912:  Procedure SetVarAsInt( const name : pastr; value : longint)');
17913:  Function IsVar( const name : pastr) : byte');
17914:  Procedure UnsetVar( const name : pastr)');
17915:  Function LineEndToBR( const s : pastr) : pastr');
17916:  Function RandomStr( len : longint) : pastr');
17917:  Function XorCrypt( const s : pastr; key : byte) : pastr');
17918:  Function CountCfgVars : longword');
17919:  Function FetchCfgVarName( idx : longword) : pastr');
17920:  Function FetchCfgVarVal( idx : longword) : pastr');
17921:  Function IsCfgVar( const name : pastr) : boolean');
17922:  Function SetCfgVar( const name, value : pastr) : boolean');
17923:  Function GetCfgVar( const name : pastr) : pastr');
17924:  Procedure ThrowErr( const s : pastr)');
17925:  Procedure ThrowWarn( const s : pastr)');
17926:  Procedure ErrWithHeader( const s : pastr)');
17927:   CL.AddTypeS('TWebVar', 'record name : pastr; value : pastr; end');
17928:   CL.AddTypeS('TWebVars', 'array of TWebVar');
17929:  Function iUpdateWebVar(var webv : TWebVars; const name, value:pastr; upcased:boolean):boolean');
17930:  Function iAddWebCfgVar( const name, value : pastr) : boolean');
17931:  Procedure iAddWebVar( var webv : TWebVars; const name, value : pastr)');
17932:  Procedure iSetRTI( const name, value : pastr)');
17933:  Function iCustomSessUnitSet : boolean');
17934:  Function iCustomCfgUnitSet : boolean');
17935: end;
17936:
17937: procedure SIRegister_W32VersionInfo(CL: TPSPascalCompiler);
17938: begin
17939:   SIRegister_TProjectVersionInfo(CL);
17940:  CL.AddDelphiFunction('Function MSLanguageToHex( const s : string) : string');
17941:  Function MSHexToLanguage( const s : string) : string');
17942:  Function MSCharacterSetToHex( const s : string) : string');
17943:  Function MSHexToCharacterSet( const s : string) : string');
17944:  Function MSLanguages : TStringList');
17945:  Function MSHexLanguages : TStringList');
17946:  Function MSCharacterSets : TStringList');
17947:  Function MSHexCharacterSets : TStringList');
17948: end;
17949:
17950: procedure SIRegister_IpUtils(CL: TPSPascalCompiler);
17951: begin
17952:   TIpHandle', 'Cardinal');
17953:   TIpMD5StateArray', 'array[0..3] of DWORD;');
17954:   CL.AddTypeS('TIpMD5CountArray', 'array[0..1] of DWORD;');
17955:   TIpMD5ByteBuf', 'array[0..63] of Byte;');
17956:   TIpMD5LongBuf', 'array[0..15] of DWORD;');
17957:   TIpMD5Digest', 'array[0..15] of Byte;');
17958:   TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther )');
17959:   TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5'
```

```
17960:    +'CountArray; ByteBuf : TIpMD5ByteBuf; end');
17961:   CL.AddClassN(CL.FindClass('TOBJECT'),'EIpBaseException');
17962:   CL.AddClassN(CL.FindClass('TOBJECT'),'EIpAccessException');
17963:   CL.AddClassN(CL.FindClass('TOBJECT'),'EIpHtmlException');
17964:   SIRegister_TIpBaseAccess(CL);
17965:   SIRegister_TIpBasePersistent(CL);
17966:   //TIpComponentClass', 'class of TIpBaseComponent');
17967:   SIRegister_TIpBaseComponent(CL);
17968:   SIRegister_TIpBaseWinControl(CL);
17969:  Function InClassA( Addr : LongInt) : Boolean');
17970:  Function InClassB( Addr : LongInt) : Boolean');
17971:  Function InClassC( Addr : LongInt) : Boolean');
17972:  Function InClassD( Addr : LongInt) : Boolean');
17973:  Function InMulticast( Addr : LongInt) : Boolean');
17974:  Function IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar) : DWORD');
17975:  Function IpCompStruct( const S1, S2, Size : Cardinal) : Integer');
17976:  Function IpMaxInt( A, B : Integer) : Integer');
17977:  Function IpMinInt( A, B : Integer) : Integer');
17978:  Procedure IpSafeFree( var Obj: TObject)');
17979:  Function IpShortVersion : string');
17980:  Function IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD) : DWORD');
17981:  Function IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD) : DWORD');
17982:  Function IpInternetSumOfFile( const FileName : string) : DWORD');
17983:  Function MD5SumOfFile( const FileName : string) : string');
17984:  Function MD5SumOfStream( Stream : TStream) : string');
17985:  Function MD5SumOfStreamDigest( Stream : TStream) : TIpMD5Digest');
17986:  Function MD5SumOfString( const S : string) : string');
17987:  Function MD5SumOfStringDigest( const S : string) : TIpMD5Digest');
17988:  Function SafeYield : LongInt');
17989:  Function AllTrimSpaces( Strng : string) : string');
17990:  Function IpCharPos( C : AnsiChar; const S : string) : Integer');
17991:  Function CharPosIdx( C : AnsiChar; const S : string; Idx : Integer) : Integer');
17992:  Function NthCharPos( C : AnsiChar; const S : string; Nth : Integer) : Integer');
17993:  Function RCharPos( C : AnsiChar; const S : string) : Integer');
17994:  Function RCharPosIdx( C : AnsiChar; const S : string; Idx : Integer) : Integer');
17995:  Function RNthCharPos( C : AnsiChar; const S : string; Nth : Integer) : Integer');
17996:  Function IpRPos( const Substr : string; const S : string) : Integer');
17997:  Function IpPosIdx( const SubStr, S : string; Idx : Integer) : Integer');
17998:   ACharSet', 'set of AnsiChar');
17999:   TIpAddrRec', 'record Scheme : string; UserName : string; Password string; Authority : string;
18000:    Port : string; Path : string; Fragment : string; Query : string; QueryDelim : AnsiChar; end');
18001:  Procedure Initialize( var AddrRec : TIpAddrRec)');
18002:  Procedure Finalize( var AddrRec : TIpAddrRec)');
18003:  Function ExtractEntityName( const NamePath : string) : string');
18004:  Function ExtractEntityPath( const NamePath : string) : string');
18005:  Function IpParseURL( const URL : string; var Rslt : TIpAddrRec) : Boolean');
18006:  Function BuildURL( const OldURL, NewURL : string) : string');
18007:  Function PutEscapes( const S : string; EscapeSet : ACharSet) : string');
18008:  Function RemoveEscapes( const S : string; EscapeSet : ACharSet) : string');
18009:  Procedure SplitParams( const Parms : string; Dest : TStrings)');
18010:  Function NetToDOSPath( const PathStr : string) : string');
18011:  Function DOSToNetPath( const PathStr : string) : string');
18012:  Procedure SplitHttpResponse( const S : string; var V, MsgID, Msg : string)');
18013:  Procedure FieldFix( Fields : TStrings)');
18014:  Function AppendSlash( APath : string) : string');
18015:  Function RemoveSlash( APath : string) : string');
18016:  Function GetParentPath( const Path : string) : string');
18017:  Function GetLocalContent( const TheFileName : string) : string');
18018:  Function IPDirExists( Dir : string) : Boolean');
18019:  Function GetTemporaryFile( const Path : string) : string');
18020:  Function GetTemporaryPath : string');
18021:  Function AppendBackSlash( APath : string) : string');
18022:  Function IpRemoveBackSlash( APath : string) : string');
18023:  Function INetDateStrToDateTime( const DateStr : string) : TDateTime');
18024:  Function DateTimeToINetDateTimeStr( DateTime : TDateTime) : string');
18025:  Function IpTimeZoneBias : Integer');
18026:  Procedure SplitCookieFields( const Data : string; Fields : TStrings)');
18027: end;
18028:
18029: procedure SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18030: begin
18031:   CL.AddTypeS('TPOStyle', '( postStandard, postPropName, postFull )');
18032:  Procedure Lrt2Po( const LRTFile : string; POStyle : TPOStyle)');
18033:  Procedure CombinePoFiles( SL : TStrings; const FName : string)');
18034: end;
18035:
18036: procedure SIRegister_GPS(CL: TPSPascalCompiler);
18037: begin
18038:  CL.AddConstantN('MAX_SATS','LongInt').SetInt( 12);
18039:  GPSMSG_START','String').SetString( '$');
18040:  GPSMSG_STOP','String').SetString( '*');
18041:  SEC_BETWEEN_SEG','LongInt').SetInt( 5);
18042:   CL.AddTypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor'
18043:    +'tint; Azimut : Smallint; SignLevel : Smallint; end');
18044:   CL.AddTypeS('TSatellites', 'array[1..12] of TSatellite');
18045:  //TSatellites = array[1..MAX_SATS] of TSatellite;
18046:   TGPSSatEvent', 'Procedure ( Sender : TObject; NbSat, NbSatUse: Shortint; Sats : TSatellites)');
18047:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Heigh'
18048:    tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid : Boolean; '
```

```
18049:    NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end');
18050:   CL.AddTypeS('TGPSDatasEvent', 'Procedure ( Sender : TObject; GPSDatas : TGPSDatas)');
18051:   CL.AddTypeS('TMsgGP', '( msgGP, msgGPGGA, msgGPGLL, msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA )');
18052:   CL.AddTypeS('TSpeedUnit', '( suKilometre, suMile, suNauticalMile )');
18053:   SIRegister_TGPSLink(CL);
18054:   SIRegister_TCustomGPS(CL);
18055:   SIRegister_TGPS(CL);
18056:   SIRegister_TGPStoGPX(CL);
18057:   SIRegister_TGPSSpeed(CL);
18058:   SIRegister_TGPSSatellitesPosition(CL);
18059:   SIRegister_TGPSSatellitesReception(CL);
18060:   SIRegister_TGPSCompass(CL);
18061:  //CL.AddDelphiFunction('Procedure Register( )');
18062:  Function IndexMsgGP( StrMsgGP : String) : TMsgGP');
18063:  Function StrCoordToAngle( Point : Char; Angle : String) : Double');
18064:  Function StrTimeToTime( const Time : String) : TDateTime');
18065:  Function StrToInteger( const Str : String) : Integer');
18066:  Function StrToReal( const Str : String) : Extended');
18067:  Function GPSRotatePoint( Angle : Double; Ct, Pt : TPoint) : TPoint');
18068:  Procedure LoadRessource( RessourceName : String; ImageList : TImageList)');
18069:  end;
18070:
18071:  procedure SIRegister_NMEA(CL: TPSPascalCompiler);
18072: begin
18073:  NMEADataArray', 'array of string');
18074:  Procedure TrimNMEA( var S : string)');
18075:  Procedure ExpandNMEA( var S : string)');
18076:  Function ParseNMEA( S : string) : NMEADataArray');
18077:  Function ChkValidNMEA( S : string) : Boolean');
18078:  Function IdNMEA( S : string) : string');
18079:  Function ChkSumNMEA( const S : string) : string');
18080:  Function PosInDeg( const PosStr : string) : Double');
18081:  Function DateTimeNMEA( const StrD, StrT : string) : TDateTime');
18082:  Function SysClockSet( const StrD, StrT : string) : Boolean');
18083:  function Ticks2Secs(Ticks : LongInt) : LongInt;');
18084:  function Secs2Ticks(Secs : LongInt) : LongInt;');
18085:  function MSecs2Ticks(MSecs : LongInt) : LongInt;');
18086: end;
18087:
18088: procedure SIRegister_SortUtils(CL: TPSPascalCompiler);
18089: begin
18090:   CL.AddTypeS('SortType1', 'Byte');
18091:   CL.AddTypeS('SortType2', 'Double');
18092:   CL.AddTypeS('SortType3', 'DWord');
18093:   //CL.AddTypeS('PDWordArray', '^DWordArray // will not work');
18094:   CL.AddTypeS('TDataRecord4', 'record Value : Integer; Data : Integer; end');
18095:  Function('Procedure QuickSort( var List : array of SortType1; Min, Max : Integer)');
18096:  Procedure QuickSortDWord( var List : array of SortType3; Min, Max : Integer)');
18097:  Procedure QuickSortDataRecord4( var List : array of TDataRecord4; Count : Integer)');
18098:  Procedure HeapSort( var List : array of SortType1; Count : DWord; FirstNeeded : DWord)');
18099:  Function QuickSelect( var List : array of SortType1; Min, Max, Wanted : Integer) : SortType1');
18100:  Function QuickSelectDouble( var List : array of SortType2; Min, Max, Wanted : Integer) : SortType2');
18101:  Function QuickSelectDWord( var List : array of SortType3; Min, Max, Wanted : Integer) : SortType3');
18102: end;
18103:
18104: procedure SIRegister_BitmapConversion(CL: TPSPascalCompiler);
18105: begin
18106:  // TMatrix3x3 = array[1..3,1..3] of Double;
18107:  // TMatrix4x4 = array[1..4,1..4] of Double;
18108:  CL.AddTypeS('TMatrix3x31', 'array[1..3] of Double');
18109:  CL.AddTypeS('TMatrix3x3', 'array[1..3] of TMatrix3x31');
18110:  CL.AddTypeS('TMatrix4x41', 'array[1..4] of Double');
18111:  CL.AddTypeS('TMatrix4x4', 'array[1..4] of TMatrix4x41');
18112:  Procedure ColorTransform( A, B, C : Byte; out X, Y, Z : Byte; const T : TMatrix4x4);');
18113:  Procedure ColorTransform1( A, B, C : Byte; out X, Y, Z : Float; const T : TMatrix4x4);');
18114:  Procedure ColorTransform2( const A, B, C : Float; out X, Y, Z : Byte; const T : TMatrix4x4);');
18115:  Procedure ColorTransformHSI2RGB( H, S, I : Byte; out R, G, B : Byte)');
18116:  Procedure ColorTransformRGB2HSI( R, G, B : Byte; out H, S, I : Byte)');
18117:  Procedure ColorTransformRGB2Lab( R, G, B : Byte; out L, a_, b_ : Byte)');
18118:  Procedure ColorTransformLab2RGB( L, a_, b_ : Byte; out R, G, B : Byte)');
18119:  Procedure ColorTransformRGB2LOCO( R, G, B : Byte; out S0, S1, S2 : Byte)');
18120:  Procedure ColorTransformLOCO2RGB( S0, S1, S2 : Byte; out R, G, B : Byte)');
18121:  Procedure ConvertColorSpace( Image : TLinarBitmap; const T : TMatrix4x4; NewImage : TLinarBitmap);');
18122:  //Procedure
       ConvertColorSpace1(Image:TLinarBitmap;ColorTransform:TColorTransformProc;NewImage:TLinarBitmap);');
18123:  Procedure ConvertToGrayscale( const Image, GrayImage : TLinarBitmap);');
18124:  Procedure ConvertToGrayscale1( const Image : TLinarBitmap);');
18125: end;
18126:
18127: procedure SIRegister_ZDbcUtils(CL: TPSPascalCompiler);
18128: begin
18129:  { TZSQLType = (zsqlstUnknown, zsqlstBoolean, zsqlstByte, zsqlstShort, zsqlstInteger, zsqlstLong,
18130:     zsqlstFloat, zsqlstDouble, zsqlstBigDecimal, zsqlstString, zsqlstUnicodeString, zsqlstBytes,
18131:     zsqlstDate, zsqlstTime, zsqlstTimestamp, zsqlstDataSet, zsqlstGUID,
18132:     stAsciiStream, stUnicodeStream, stBinaryStream);}
18133:  Function ResolveConnectionProtocol( Url : string; SupportedProtocols : TStringDynArray) : string');
18134:  //Procedure ResolveDatabaseUrl( const Url: string; Info:TStrings; var HostName:string; var
       Port:Integer;var Database : string; var UserName : string; var Password : string; ResultInfo : TStrings)');
18135:  Function CheckConvertion( InitialType : TZSQLType; ResultType : TZSQLType) : Boolean');
```

```
18136:  Function DefineColumnTypeName( ColumnType : TZSQLType) : string');
18137:  Procedure RaiseSQLException( E : Exception)');
18138:  //Procedure CopyColumnsInfo( FromList : TObjectList; ToList : TObjectList)');
18139:  Function ToLikeString( const Value : string) : string');
18140:  Function GetSQLHexWideString( Value : PChar; Len : Integer; ODBC : Boolean) : WideString');
18141:  Function GetSQLHexAnsiString( Value : PChar; Len : Integer; ODBC : Boolean) : RawByteString');
18142:  Function GetSQLHexString( Value : PChar; Len : Integer; ODBC : Boolean) : String');
18143:  Function WideStringStream( const AString : WideString) : TStream');
18144:  function ConvertAdoToTypeName(FieldType: SmallInt): string;
18145:  function GetTableName(const AField: TField): string;');
18146:  function GetFieldName(const AField: TField): string;');
18147: end;
18148:
18149: procedure SIRegister_JclTD32(CL: TPSPascalCompiler);
18150:   CL.AddTypeS('TSymbolInfo', 'record Size : Word; SymbolType : Word; end');
18151:   //CL.AddTypeS('PSymbolInfos', '^TSymbolInfos // will not work');
18152:   SIRegister_TJclModuleInfo(CL);
18153:   SIRegister_TJclLineInfo(CL);
18154:   SIRegister_TJclSourceModuleInfo(CL);
18155:   SIRegister_TJclSymbolInfo(CL);
18156:   SIRegister_TJclProcSymbolInfo(CL);
18157:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclLocalProcSymbolInfo');
18158:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclGlobalProcSymbolInfo');
18159:   SIRegister_TJclTD32InfoParser(CL);
18160:   SIRegister_TJclTD32InfoScanner(CL);
18161:   SIRegister_TJclPeBorTD32Image(CL);
18162: end;
18163:
18164: procedure SIRegister_JvIni(CL: TPSPascalCompiler);
18165: begin
18166:   CL.AddTypeS('TReadObjectEvent', 'Function ( Sender : TObject; const Section, '
18167:    +'Item, Value : string) : TObject');
18168:   TWriteObjectEvent', 'Procedure ( Sender : TObject; const Section, Item: string; Obj: TObject)');
18169:  Function StringToFontStyles( const Styles : string) : TFontStyles');
18170:  Function FontStylesToString( Styles : TFontStyles) : string');
18171:  Function FontToString( Font : TFont) : string');
18172:  Procedure StringToFont( const Str : string; Font : TFont)');
18173:  Function RectToStr( Rect : TRect) : string');
18174:  Function StrToRect( const Str : string; const Def : TRect) : TRect');
18175:  Function JPointToStr( P : TPoint) : string');
18176:  Function JStrToPoint( const Str : string; const Def : TPoint) : TPoint');
18177:  Function DefProfileName : string');
18178:  Function DefLocalProfileName : string');
18179:  CL.AddConstantN('idnListItem','String').SetString( 'Item');
18180: end;
18181:
18182: procedure SIRegister_JvHtControls(CL: TPSPascalCompiler);
18183: begin
18184:  Procedure ItemHtDrawEx( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text :
       string; const HideSelColor : Boolean; var PlainItem : string; var Width : Integer; CalcWidth : Boolean)');
18185:  Function ItemHtDraw( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string;
       const HideSelColor : Boolean) : string');
18186:  Function ItemHtWidth( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string;
       const HideSelColor : Boolean) : Integer');
18187:  Function ItemHtPlain( const Text : string) : string');
18188:  Procedure ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:string);
18189:  Function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string;MouseX,MouseY:Integer;var
       HyperLink:string):Bool;
18190: end;
18191:
18192: procedure SIRegister_NeuralNetwork(CL: TPSPascalCompiler);
18193: begin
18194:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNeuron');
18195:   CL.AddTypeS('TSynapse', 'record W : Real; Connection : TNeuron; end');
18196:   CL.AddTypeS('TAcson', 'record Alfa : Real; Beta : Real; Gama : Real; end');
18197:   SIRegister_TNeuron(CL);
18198:   SIRegister_TNeuronLayer(CL);
18199:   SIRegister_TNeuralNet(CL);
18200: end;
18201:
18202: procedure SIRegister_StExpr(CL: TPSPascalCompiler);
18203: begin
18204:   //CL.AddTypeS('PStFloat', '^TStFloat // will not work');
18205:   TStMethod0Param', 'Function  : TStFloat');
18206:   TStMethod1Param', 'Function ( Value1 : TStFloat) : TStFloat');
18207:   TStMethod2Param', 'Function ( Value1, Value2 : TStFloat) : TStFloat');
18208:   TStMethod3Param', 'Function ( Value1, Value2, Value3 : TStFloat) : TStFloat');
18209:   TStGetIdentValueEvent', 'Procedure ( Sender : TObject; const Ide'
18210:    +'ntifier : AnsiString; var Value : TStFloat)');
18211:   TStToken', '( ssStart, ssInIdent, ssInNum, ssInSign, ssInExp, ss'
18212:    +'Eol, ssNum, ssIdent, ssLPar, ssRPar, ssComma, ssPlus, ssMinus, ssTimes, ssDiv, ssEqual, ssPower )');
18213:   SIRegister_TStExpression(CL);
18214:   TStExprErrorEvent', 'Procedure ( Sender : TObject; ErrorNumber :'
18215:    +' LongInt; const ErrorStr : AnsiString)');
18216:   SIRegister_TStExpressionEdit(CL);
18217:  Function AnalyzeExpr( const Expr : AnsiString) : Double');
18218:  Procedure TpVal( const S : AnsiString; var V : Extended; var Code : Integer)');
18219: end;
18220:
```

```
18221: procedure SIRegister_GR32_Containers(CL: TPSPascalCompiler);
18222: begin
18223:  CL.AddConstantN('BUCKET_MASK','LongWord').SetUInt( $FF);
18224:  CL.AddConstantN('BUCKET_COUNT','LongInt').SetInt( BUCKET_MASK + 1);
18225:  Procedure SmartAssign( Src, Dst : TPersistent; TypeKinds : TTypeKinds)');
18226:  Procedure Advance( var Node : TLinkedNode; Steps : Integer)');
18227: end;
18228:
18229: procedure SIRegister_StSaturn(CL: TPSPascalCompiler);
18230: begin
18231:  TStJupSatPos', 'record X: double; Y: Double; end');
18232:  TStJupSats', 'record Io: TStJupSatPos;
       Europa:TStJupSatPos;Ganymede:TStJupSatPos;Callisto:TStJupSatPos;end;
18233:  Function ComputeSaturn( JD : Double) : TStEclipticalCord');
18234:  Function ComputePluto( JD : Double) : TStEclipticalCord');
18235:  Function ComputeVenus( JD : Double) : TStEclipticalCord');
18236:  Function ComputeMars( JD : Double) : TStEclipticalCord');
18237:  Function ComputeMercury( JD : Double) : TStEclipticalCord');
18238:  Function ComputeJupiter( JD : Double) : TStEclipticalCord');
18239:  Function ComputeUranus( JD : Double) : TStEclipticalCord');
18240:  Function ComputeNeptune( JD : Double) : TStEclipticalCord');
18241:  function GetJupSats(JD : TDateTime; HighPrecision, Shadows : Boolean) : TStJupSats;');
18242: end;
18243:
18244: procedure SIRegister_JclParseUses(CL: TPSPascalCompiler);
18245: begin
18246:   CL.AddClassN(CL.FindClass('TOBJECT'),'EUsesListError');
18247:  Function CreateGoal( Text : PChar) : TCustomGoal');
18248: end;
18249:
18250: procedure SIRegister_JvFinalize(CL: TPSPascalCompiler);
18251: begin
18252: //type
18253: //  TFinalizeProc = procedure;
18254:   CL.AddTypeS('TFinalizeProc', 'procedure');
18255:  Procedure AddFinalizeProc( const UnitName : string; FinalizeProc : TFinalizeProc)');
18256:  Function AddFinalizeObject( const UnitName : string; Instance : TObject) : TObject');
18257:  Function AddFinalizeObjectNil( const UnitName : string; var Reference: TObject) : TObject');
18258:  Function AddFinalizeFreeAndNil( const UnitName : string; var Reference: TObject) : TObject');
18259:  Function AddFinalizeMemory( const UnitName : string; Ptr : ___Pointer) : ___Pointer');
18260:  Function AddFinalizeMemoryNil( const UnitName : string; var Ptr: ___Pointer) : ___Pointer');
18261:  Procedure FinalizeUnit( const UnitName : string)');
18262: end;
18263:
18264:
18265: function TRestRequest_createStringStreamFromStringList(strings: TStringList): TStringStream;
18266:
18267: {A simple Oscilloscope using  TWaveIn class.
18268:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
18269:  http://www.retroarchive.org/garbo/pc/turbopas/index.html
18270: uses
18271:   Forms,
18272:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
18273:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
18274:   uColorFunctions in 'uColorFunctions.pas',
18275:   AMixer in 'AMixer.pas',
18276:   uSettings in 'uSettings.pas',
18277:   UWavein4 in 'UWavein4.pas',
18278:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
18279:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
18280:
18281: Functions_max hex in the box maXbox
18282: functionslist.txt
18283: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98/100/101/110/120
18284:
18285: ********************************************************************************
18286: Procedure
18287: PROCEDURE SIZE 8516 8396 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
18288: Procedure ***************************Now the Procedure list******************
18289: Procedure ( ACol, ARow : Integer; Items : TStrings)
18290: Procedure ( Agg : TAggregate)
18291: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
18292: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
18293: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
18294: Procedure ( ASender : TObject; const ABytes : Integer)
18295: Procedure ( ASender : TObject; VStream : TStream)
18296: Procedure ( AThread : TIdThread)
18297: Procedure ( AWebModule : TComponent)
18298: Procedure ( Column : TColumn)
18299: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
18300: Procedure ( const iStart : integer; const sText : string)
18301: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
18302: Procedure ( Database : TDatabase; LoginParams : TStrings)
18303: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
18304: Procedure ( DATASET : TDATASET)
18305: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
18306: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
18307: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
```

```
18308: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
18309: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
18310: Procedure ( Done : Integer)
18311: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
18312: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
18313: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
18314: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
18315: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
18316: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
18317: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
18318: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
18319: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
18320: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
18321: Procedure ( SENDER : TFIELD; const TEXT : String)
18322: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
18323: Procedure ( Sender : TIdTelnet; const Buffer : String)
18324: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
18325: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
18326: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18327: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
18328: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
18329: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
18330: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
18331: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
18332: Procedure ( Sender : TObject; Button : TMPBtnType)
18333: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
18334: Procedure ( Sender : TObject; Button : TUDBtnType)
18335: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
18336: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
18337: Procedure ( Sender : TObject; Column : TListColumn)
18338: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
18339: Procedure ( Sender : TObject; Connecting : Boolean)
18340: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
18341: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
18342: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
18343: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
18344: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
18345: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
18346: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
18347: Procedure ( Sender : TObject; Index : LongInt)
18348: Procedure ( Sender : TObject; Item : TListItem)
18349: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
18350: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
18351: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
18352: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
18353: Procedure ( Sender : TObject; Item : TListItem; var S : string)
18354: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
18355: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
18356: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
18357: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
18358: Procedure ( Sender : TObject; Node : TTreeNode)
18359: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
18360: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
18361: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
18362: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
18363: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
18364: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
18365: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
18366: Procedure ( Sender : TObject; Rect : TRect)
18367: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
18368: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
18369: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
18370: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
18371: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
18372: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
18373: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
18374: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
18375: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
18376: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
18377: Procedure ( Sender : TObject; Thread : TServerClientThread)
18378: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
18379: Procedure ( Sender : TObject; Username, Password : string)
18380: Procedure ( Sender : TObject; var AllowChange : Boolean)
18381: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
18382: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
18383: Procedure ( Sender : TObject; var Continue : Boolean)
18384: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
18385: Procedure ( Sender : TObject; var Username : string)
18386: Procedure ( Sender : TObject; Wnd : HWND)
18387: Procedure ( Sender : TToolbar; Button : TToolButton)
18388: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
18389: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
18390: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
18391: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
18392: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
18393: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
18394: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
18395: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
```

```
18396: procedure (Sender: TObject)
18397: procedure (Sender: TObject; var Done: Boolean)
18398: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
18399: procedure _T(Name: tbtString; v: Variant);
18400: Procedure AbandonSignalHandler( RtlSigNum : Integer)
18401: Procedure Abort
18402: Procedure About1Click( Sender : TObject)
18403: Procedure Accept( Socket : TSocket)
18404: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
18405: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
18406: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
18407: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
18408: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
18409: Procedure Add( Addend1, Addend2 : TMyBigInt)
18410: Procedure ADD( const AKEY, AVALUE : VARIANT)
18411: Procedure Add( const Key : string; Value : Integer)
18412: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
18413: Procedure ADD( FIELD : TFIELD)
18414: Procedure ADD( ITEM : TMENUITEM)
18415: Procedure ADD( POPUP : TPOPUPMENU)
18416: Procedure AddCharacters( xCharacters : TCharSet)
18417: Procedure AddDriver( const Name : string; List : TStrings)
18418: Procedure AddImages( Value : TCustomImageList)
18419: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
18420: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
18421: Procedure AddLoader( Loader : TBitmapLoader)
18422: Procedure ADDPARAM( VALUE : TPARAM)
18423: Procedure AddPassword( const Password : string)
18424: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
18425: Procedure AddState( oState : TniRegularExpressionState)
18426: Procedure AddStrings( Strings : TStrings);
18427: procedure AddStrings(Strings: TStrings);
18428: Procedure AddStrings1( Strings : TWideStrings);
18429: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
18430: Procedure AddToRecentDocs( const Filename : string)
18431: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
18432: Procedure AllFunctionsList1Click( Sender : TObject)
18433: procedure AllObjectsList1Click(Sender: TObject);
18434: Procedure Allocate( AAllocateBytes : Integer)
18435: procedure AllResourceList1Click(Sender: TObject);
18436: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
18437: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
18438: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
18439: Procedure AnsiFree( var s : AnsiString)
18440: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
18441: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
18442: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
18443: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
18444: Procedure AntiFreeze;
18445: Procedure APPEND
18446: Procedure Append( const S : WideString)
18447: procedure Append(S: string);
18448: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
18449: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
18450: Procedure AppendChunk( Val : OleVariant)
18451: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
18452: Procedure AppendStr( var Dest : string; S : string)
18453: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
18454: Procedure ApplyRange
18455: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18456: Procedure Arrange( Code : TListArrangement)
18457: procedure Assert(expr : Boolean; const msg: string);
18458: procedure Assert2(expr : Boolean; const msg: string);
18459: Procedure Assign( AList : TCustomBucketList)
18460: Procedure Assign( Other : TObject)
18461: Procedure Assign( Source : TDragObject)
18462: Procedure Assign( Source : TPersistent)
18463: Procedure Assign(Source: TPersistent)
18464: procedure Assign2(mystring, mypath: string);
18465: Procedure AssignCurValues( Source : TDataSet);
18466: Procedure AssignCurValues1( const CurValues : Variant);
18467: Procedure ASSIGNFIELD( FIELD : TFIELD)
18468: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
18469: Procedure AssignFile(var F: Text; FileName: string)
18470: procedure AssignFile(var F: TextFile; FileName: string)
18471: procedure AssignFileRead(var mystring, myfilename: string);
18472: procedure AssignFileWrite(mystring, myfilename: string);
18473: Procedure AssignTo( Other : TObject)
18474: Procedure AssignValues( Value : TParameters)
18475: Procedure ASSIGNVALUES( VALUE : TPARAMS)
18476: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
18477: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
18478: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
18479: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
18480: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18481: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
18482: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
18483: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
18484: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
```

```
18485: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
18486: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
18487: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
18488: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
18489: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18490: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
18491: procedure Beep
18492: Procedure BeepOk
18493: Procedure BeepQuestion
18494: Procedure BeepHand
18495: Procedure BeepExclamation
18496: Procedure BeepAsterisk
18497: Procedure BeepInformation
18498: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
18499: Procedure BeginLayout
18500: Procedure BeginTimer( const Delay, Resolution : Cardinal)
18501: Procedure BeginUpdate
18502: procedure BeginUpdate;
18503: procedure BigScreen1Click(Sender: TObject);
18504: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
18505: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
18506: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
18507: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
18508: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
18509: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
18510: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
18511: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
18512: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
18513: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
18514: Procedure BreakPointMenuClick( Sender : TObject)
18515: procedure BRINGTOFRONT
18516: procedure BringToFront;
18517: Procedure btnBackClick( Sender : TObject)
18518: Procedure btnBrowseClick( Sender : TObject)
18519: Procedure BtnClick( Index : TNavigateBtn)
18520: Procedure btnLargeIconsClick( Sender : TObject)
18521: Procedure BuildAndSendRequest( AURI : TIdURI)
18522: Procedure BuildCache
18523: Procedure BurnMemory( var Buff, BuffLen : integer)
18524: Procedure BurnMemoryStream( Destructo : TMemoryStream)
18525: Procedure CalculateFirstSet
18526: Procedure Cancel
18527: procedure CancelDrag;
18528: Procedure CancelEdit
18529: procedure CANCELHINT
18530: Procedure CancelRange
18531: Procedure CancelUpdates
18532: Procedure CancelWriteBuffer
18533: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
18534: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
18535: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
18536: procedure CaptureScreenFormat(vname: string; vextension: string);
18537: procedure CaptureScreenPNG(vname: string);
18538: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
18539: procedure CASCADE
18540: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
18541: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
18542: Procedure cbPathClick( Sender : TObject)
18543: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18544: Procedure cedebugAfterExecute( Sender : TPSScript)
18545: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18546: Procedure cedebugCompile( Sender : TPSScript)
18547: Procedure cedebugExecute( Sender : TPSScript)
18548: Procedure cedebugIdle( Sender : TObject)
18549: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18550: Procedure CenterHeight( const pc, pcParent : TControl)
18551: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18552: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18553: Procedure Change
18554: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18555: Procedure Changed
18556: Procedure ChangeDir( const ADirName : string)
18557: Procedure ChangeDirUp
18558: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
18559: Procedure ChangeLevelBy( Value : TChangeRange)
18560: Procedure ChDir(const s: string)
18561: Procedure Check(Status: Integer)
18562: Procedure CheckCommonControl( CC : Integer)
18563: Procedure CHECKFIELDNAME( const FIELDNAME : String)
18564: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
18565: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
18566: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
18567: Procedure CheckToken( T : Char)
18568: procedure CheckToken(t:char)
18569: Procedure CheckTokenSymbol( const S : string)
18570: procedure CheckTokenSymbol(s:string)
18571: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
18572: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18573: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
```

```
18574: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
18575: procedure CipherFile1Click(Sender: TObject);
18576: Procedure Clear;
18577: Procedure Clear1Click( Sender : TObject)
18578: Procedure ClearColor( Color : TColor)
18579: Procedure CLEARITEM( AITEM : TMENUITEM)
18580: Procedure ClearMapping
18581: Procedure ClearSelection( KeepPrimary : Boolean)
18582: Procedure ClearWriteBuffer
18583: Procedure Click
18584: Procedure Close
18585: Procedure Close1Click( Sender : TObject)
18586: Procedure CloseDatabase( Database : TDatabase)
18587: Procedure CloseDataSets
18588: Procedure CloseDialog
18589: Procedure CloseFile(var F: Text);
18590: Procedure Closure
18591: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18592: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18593: Procedure CodeCompletionList1Click( Sender : TObject)
18594: Procedure ColEnter
18595: Procedure Collapse
18596: Procedure Collapse( Recurse : Boolean)
18597: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
18598: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
18599: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
18600: Procedure Compile1Click( Sender : TObject)
18601: procedure ComponentCount1Click(Sender: TObject);
18602: Procedure Compress(azipfolder, azipfile: string)
18603: Procedure DeCompress(azipfolder, azipfile: string)
18604: Procedure XZip(azipfolder, azipfile: string)
18605: Procedure XUnZip(azipfolder, azipfile: string)
18606: Procedure Connect( const ATimeout : Integer)
18607: Procedure Connect( Socket : TSocket)
18608: procedure Console1Click(Sender: TObject);
18609: Procedure Continue
18610: Procedure ContinueCount( var Counter : TJclCounter)
18611: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18612: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
18613: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
18614: Procedure ConvertImage(vsource, vdestination: string);
18615: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
18616: Procedure ConvertBitmap(vsource, vdestination: string);
18617: Procedure ConvertToGray(Cnv: TCanvas);
18618: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
18619: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
18620: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
18621: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
18622: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
18623: Procedure CopyFrom( mbCopy : TMyBigInt)
18624: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
18625: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
18626: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
18627: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
18628: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
18629: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
18630: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
18631: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
18632: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
18633: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18634: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
18635: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18636: Procedure CopyToClipboard
18637: Procedure CountParts
18638: Procedure CreateDataSet
18639: Procedure CreateEmptyFile( const FileName : string)
18640: Procedure CreateFileFromString( const FileName, Data : string)
18641: Procedure CreateFromDelta( Source : TPacketDataSet)
18642: procedure CREATEHANDLE
18643: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18644: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
18645: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
18646: Procedure CreateTable
18647: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
18648: procedure CSyntax1Click(Sender: TObject);
18649: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
18650: Procedure CURSORPOSCHANGED
18651: procedure CutFirstDirectory(var S: String)
18652: Procedure DataBaseError(const Message: string)
18653: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
18654: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
18655: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
18656: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18657: Procedure DBIError(errorCode: Integer)
18658: Procedure DebugOutput( const AText : string)
18659: procedure Debugln(DebugLOGFILE: string; Event_message: string);
```

```
18660: Procedure DebugRun1Click( Sender : TObject)
18661: procedure Dec;
18662: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
18663: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
18664: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
18665: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
18666: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
18667: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
18668: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18669: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
18670: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
18671: Procedure Decompile1Click( Sender : TObject)
18672: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18673: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18674: Procedure DeferLayout
18675: Procedure defFileread
18676: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
18677: Procedure DelayMicroseconds( const MicroSeconds : Integer)
18678: Procedure Delete
18679: Procedure Delete( const AFilename : string)
18680: Procedure Delete( const Index : Integer)
18681: Procedure DELETE( INDEX : INTEGER)
18682: Procedure Delete( Index : LongInt)
18683: Procedure Delete( Node : TTreeNode)
18684: procedure Delete(var s: AnyString; ifrom, icount: Longint);
18685: Procedure DeleteAlias( const Name : string)
18686: Procedure DeleteDriver( const Name : string)
18687: Procedure DeleteIndex( const Name : string)
18688: Procedure DeleteKey( const Section, Ident : String)
18689: Procedure DeleteRecords
18690: Procedure DeleteRecords( AffectRecords : TAffectRecords)
18691: Procedure DeleteString( var pStr : String; const pDelStr : string)
18692: Procedure DeleteTable
18693: procedure DelphiSite1Click(Sender: TObject);
18694: Procedure Deselect
18695: Procedure Deselect( Node : TTreeNode)
18696: procedure DestroyComponents
18697: Procedure DestroyHandle
18698: Procedure Diff( var X : array of Double)
18699: procedure Diff(var X: array of Double);
18700: Procedure DirCreate( const DirectoryName : String)');
18701: procedure DISABLEALIGN
18702: Procedure DisableConstraints
18703: Procedure Disconnect
18704: Procedure Disconnect( Socket : TSocket)
18705: Procedure Dispose
18706: procedure Dispose(P: PChar)
18707: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
18708: Procedure DoKey( Key : TDBCtrlGridKey)
18709: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18710: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18711: Procedure Dormant
18712: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
18713: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
18714: Procedure DoubleToComp( Value : Double; var Result : Comp)
18715: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18716: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
18717: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18718: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18719: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18720: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
18721: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18722: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
18723: procedure DrawFocusRect(const Rect: TRect);
18724: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
18725: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18726: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
18727: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
18728: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18729: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
18730: Procedure DropConnections
18731: Procedure DropDown
18732: Procedure DumpDescription( oStrings : TStrings)
18733: Procedure DumpStateTable( oStrings : TStrings)
18734: Procedure EDIT
18735: Procedure EditButtonClick
18736: Procedure EditFont1Click( Sender : TObject)
18737: procedure Ellipse(X1, Y1, X2, Y2: Integer);
18738: Procedure Ellipse1( const Rect : TRect);
18739: Procedure EMMS
18740: Procedure Encode( ADest : TStream)
18741: procedure ENDDRAG(DROP:BOOLEAN)
18742: Procedure EndEdit( Cancel : Boolean)
18743: Procedure EndTimer
18744: Procedure EndUpdate
18745: Procedure EraseSection( const Section : string)
18746: Procedure Error( const Ident : string)
```

```
18747: procedure Error(Ident:Integer)
18748: Procedure ErrorFmt( const Ident : string; const Args : array of const)
18749: Procedure ErrorStr( const Message : string)
18750: procedure ErrorStr(Message:String)
18751: Procedure Exchange( Index1, Index2 : Integer)
18752: procedure Exchange(Index1, Index2: Integer);
18753: Procedure Exec( FileName, Parameters, Directory : string)
18754: Procedure ExecProc
18755: Procedure ExecSQL( UpdateKind : TUpdateKind)
18756: Procedure Execute
18757: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
18758: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
18759: Procedure ExecuteCommand(executeFile, paramstring: string)
18760: Procedure ExecuteShell(executeFile, paramstring: string)
18761: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18762: Procedure ExitThread(ExitCode: Integer); stdcall;
18763: Procedure ExitProcess(ExitCode: Integer); stdcall;
18764: Procedure Expand( AUserName : String; AResults : TStrings)
18765: Procedure Expand( Recurse : Boolean)
18766: Procedure ExportClipboard1Click( Sender : TObject)
18767: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
18768: Procedure ExtractContentFields( Strings : TStrings)
18769: Procedure ExtractCookieFields( Strings : TStrings)
18770: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
18771: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18772: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18773: Procedure ExtractQueryFields( Strings : TStrings)
18774: Procedure FastDegToGrad
18775: Procedure FastDegToRad
18776: Procedure FastGradToDeg
18777: Procedure FastGradToRad
18778: Procedure FastRadToDeg
18779: Procedure FastRadToGrad
18780: Procedure FileClose( Handle : Integer)
18781: Procedure FileClose(handle: integer)
18782: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18783: Procedure FileStructure( AStructure : TIdFTPDataStructure)
18784: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
18785: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
18786: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
18787: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18788: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18789: Procedure FillIPList
18790: procedure FillRect(const Rect: TRect);
18791: Procedure FillTStrings( AStrings : TStrings)
18792: Procedure FilterOnBookmarks( Bookmarks : array of const)
18793: procedure FinalizePackage(Module: HMODULE)
18794: procedure FindClose;
18795: procedure FindClose2(var F: TSearchRec)
18796: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
18797: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
18798: Procedure FindNearest( const KeyValues : array of const)
18799: Procedure FinishContext
18800: Procedure FIRST
18801: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
18802: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18803: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
18804: Procedure FlushSchemaCache( const TableName : string)
18805: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
18806: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
18807: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
18808: Procedure FormActivate( Sender : TObject)
18809: procedure FormatLn(const format: String; const args: array of const);  //alias
18810: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18811: Procedure FormCreate( Sender : TObject)
18812: Procedure FormDestroy( Sender : TObject)
18813: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18814: procedure FormOutput1Click(Sender: TObject);
18815: Procedure FormToHtml( Form : TForm; Path : string)
18816: procedure FrameRect(const Rect: TRect);
18817: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18818: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
18819: Procedure Free( Buffer : TRecordBuffer)
18820: Procedure Free( Buffer : TValueBuffer)
18821: Procedure Free;
18822: Procedure FreeAndNil(var Obj:TObject)
18823: Procedure FreeImage
18824: procedure FreeMem(P: PChar; Size: Integer)
18825: Procedure FreeTreeData( Tree : TUpdateTree)
18826: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
18827: Procedure FullCollapse
18828: Procedure FullExpand
18829: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
18830: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
18831: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
18832: Procedure Get1( AURL : string; const AResponseContent : TStream);
18833: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
```

```
18834: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
18835: Procedure GetAliasNames( List : TStrings)
18836: Procedure GetAliasParams( const AliasName : string; List : TStrings)
18837: Procedure GetApplicationsRunning( Strings : TStrings)
18838: Procedure getBox(aURL, extension: string);
18839: Procedure GetCommandTypes( List : TWideStrings)
18840: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
18841: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
18842: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
18843: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
18844: Procedure GetDatabaseNames( List : TStrings)
18845: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
18846: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
18847: Procedure GetDir(d: byte; var s: string)
18848: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
18849: Procedure GetDriverNames( List : TStrings)
18850: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
18851: Procedure GetDriverParams( const DriverName : string; List : TStrings)
18852: Procedure GetEMails1Click( Sender : TObject)
18853: Procedure getEnvironmentInfo;
18854: Function getEnvironmentString: string;
18855: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
18856: Procedure GetFieldNames( const TableName : string; List : TStrings)
18857: Procedure GetFieldNames( const TableName : string; List : TStrings);
18858: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
18859: Procedure GETFIELDNAMES( LIST : TSTRINGS)
18860: Procedure GetFieldNames1( const TableName : string; List : TStrings);
18861: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
18862: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
18863: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
18864: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
18865: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
18866: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
18867: Procedure GetFormatSettings
18868: Procedure GetFromDIB( var DIB : TBitmapInfo)
18869: Procedure GetFromHDIB( HDIB : HBitmap)
18870: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
18871: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
18872: Procedure GetIcon( Index : Integer; Image : TIcon);
18873: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
18874: Procedure GetIndexInfo( IndexName : string)
18875: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
18876: Procedure GetIndexNames( List : TStrings)
18877: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
18878: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
18879: Procedure GetIndexNames4( const TableName : string; List : TStrings);
18880: Procedure GetInternalResponse
18881: Procedure GETITEMNAMES( LIST : TSTRINGS)
18882: procedure GetMem(P: PChar; Size: Integer)
18883: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
18884: procedure GetPackageDescription(ModuleName: PChar): string)
18885: Procedure GetPackageNames( List : TStrings);
18886: Procedure GetPackageNames1( List : TWideStrings);
18887: Procedure GetParamList( List : TList; const ParamNames : WideString)
18888: Procedure GetProcedureNames( List : TStrings);
18889: Procedure GetProcedureNames( List : TWideStrings);
18890: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
18891: Procedure GetProcedureNames1( List : TStrings);
18892: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
18893: Procedure GetProcedureNames3( List : TWideStrings);
18894: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
18895: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
18896: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
18897: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
18898: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
18899: Procedure GetProviderNames( Names : TWideStrings);
18900: Procedure GetProviderNames( Proc : TGetStrProc)
18901: Procedure GetProviderNames1( Names : TStrings);
18902: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
18903: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
18904: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
18905: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
18906: Procedure GetSchemaNames( List : TStrings);
18907: Procedure GetSchemaNames1( List : TWideStrings);
18908: Procedure getScriptandRunAsk;
18909: Procedure getScriptandRun(ascript: string);
18910: Procedure getScript(ascript: string);   //alias
18911: Procedure getWebScript(ascript: string);   //alias
18912: Procedure GetSessionNames( List : TStrings)
18913: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
18914: Procedure GetStrings( List : TStrings)
18915: Procedure GetSystemTime; stdcall;
18916: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
18917: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
18918: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
18919: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
18920: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
18921: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
```

```
18922: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
18923: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
18924: Procedure GetVisibleWindows( List : Tstrings)
18925: Procedure GoBegin
18926: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
18927: Procedure GotoCurrent( Table : TTable)
18928: procedure GotoEnd1Click(Sender: TObject);
18929: Procedure GotoNearest
18930: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
18931: Procedure HandleException( E : Exception; var Handled : Boolean)
18932: procedure HANDLEMESSAGE
18933: procedure HandleNeeded;
18934: Procedure Head( AURL : string)
18935: Procedure Help( var AHelpContents : TStringList; ACommand : String)
18936: Procedure HexToBinary( Stream : TStream)
18937: procedure HexToBinary(Stream:TStream)
18938: Procedure HideDragImage
18939: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
18940: Procedure HideTraybar
18941: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18942: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18943: Procedure HookOSExceptions
18944: Procedure HookSignal( RtlSigNum : Integer)
18945: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
18946: Procedure HTMLSyntax1Click( Sender : TObject)
18947: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18948: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
18949: Procedure ImportfromClipboard1Click( Sender : TObject)
18950: Procedure ImportfromClipboard2Click( Sender : TObject)
18951: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
18952: procedure Incb(var x: byte);
18953: Procedure Include1Click( Sender : TObject)
18954: Procedure IncludeOFF;  //preprocessing
18955: Procedure IncludeON;
18956: procedure Info1Click(Sender: TObject);
18957: Procedure InitAltRecBuffers( CheckModified : Boolean)
18958: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18959: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
18960: Procedure InitData( ASource : TDataSet)
18961: Procedure InitDelta( ADelta : TPacketDataSet);
18962: Procedure InitDelta1( const ADelta : OleVariant);
18963: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18964: Procedure Initialize
18965: procedure InitializePackage(Module: HMODULE)
18966: Procedure INITIATEACTION
18967: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
18968: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18969: Procedure InitModule( AModule : TComponent)
18970: Procedure InitStdConvs
18971: Procedure InitTreeData( Tree : TUpdateTree)
18972: Procedure INSERT
18973: Procedure Insert( Index : Integer; AClass : TClass)
18974: Procedure Insert( Index : Integer; AComponent : TComponent)
18975: Procedure Insert( Index : Integer; AObject : TObject)
18976: Procedure Insert( Index : Integer; const S : WideString)
18977: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18978: Procedure Insert(Index: Integer; const S: string);
18979: procedure Insert(Index: Integer; S: string);
18980: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18981: procedure InsertComponent(AComponent:TComponent)
18982: procedure InsertControl(AControl: TControl);
18983: Procedure InsertIcon( Index : Integer; Image : TIcon)
18984: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
18985: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18986: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18987: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18988: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18989: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18990: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
18991: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18992: Procedure InvalidateModuleCache
18993: Procedure InvalidateTitles
18994: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18995: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18996: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
18997: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18998: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18999: procedure JavaSyntax1Click(Sender: TObject);
19000: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
19001: Procedure KillDataChannel
19002: Procedure Largefont1Click( Sender : TObject)
19003: Procedure LAST
19004: Procedure LaunchCpl( FileName : string)
19005: Procedure Launch( const AFile : string)
19006: Procedure LaunchFile( const AFile : string)
19007: Procedure LetFileList(FileList: TStringlist; apath: string);
19008: Procedure lineToNumber( xmemo : String; met : boolean)
```

```
19009: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
19010: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
19011: Procedure ListViewData( Sender : TObject; Item : TListItem)
19012: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
19013: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
19014: Procedure ListViewDblClick( Sender : TObject)
19015: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
19016: Procedure ListDLLExports(const FileName: string; List: TStrings);
19017: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
19018: procedure LoadBytecode1Click(Sender: TObject);
19019: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
19020: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
19021: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
19022: Procedure LoadFromFile( AFileName : string)
19023: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
19024: Procedure LoadFromFile( const FileName : string)
19025: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
19026: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
19027: Procedure LoadFromFile( const FileName : WideString)
19028: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19029: Procedure LoadFromFile(const AFileName: string)
19030: procedure LoadFromFile(FileName: string);
19031: procedure LoadFromFile(FileName:String)
19032: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
19033: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
19034: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
19035: Procedure LoadFromStream( const Stream : TStream)
19036: Procedure LoadFromStream( S : TStream)
19037: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
19038: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19039: Procedure LoadFromStream( Stream : TStream)
19040: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
19041: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
19042: procedure LoadFromStream(Stream: TStream);
19043: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
19044: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
19045: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
19046: Procedure LoadLastFile1Click( Sender : TObject)
19047: { LoadIcoToImage loads two icons from resource named NameRes,
19048:   into two image lists ALarge and ASmall}
19049: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
19050: Procedure LoadMemo
19051: Procedure LoadParamsFromIniFile( FFileName : WideString)
19052: Procedure Lock
19053: Procedure Login
19054: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
19055: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
19056: Procedure MakeCaseInsensitive
19057: Procedure MakeDeterministic( var bChanged : boolean)
19058: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
19059: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
19060: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
19061: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
19062: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19063: Procedure SetRectComplexFormatStr( const S : string)
19064: Procedure SetPolarComplexFormatStr( const S : string)
19065: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19066: Procedure MakeVisible
19067: Procedure MakeVisible( PartialOK : Boolean)
19068: Procedure Manual1Click( Sender : TObject)
19069: Procedure MarkReachable
19070: Procedure maXbox;  //shows the exe version data in a win box
19071: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
19072: Procedure Memo1Change( Sender : TObject)
19073: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
19074: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
19075: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
19076: procedure Memory1Click(Sender: TObject);
19077: Procedure MERGE( MENU : TMAINMENU)
19078: Procedure MergeChangeLog
19079: procedure MINIMIZE
19080: Procedure MinimizeMaxbox;
19081: procedure MyCopyFile(Name1,Name2:string);
19082: Procedure MkDir(const s: string)
19083: Procedure MakeDir(const s: string)');
19084: Procedure ChangeDir(const s: string)');
19085: Function makeFile(const FileName: string): integer)');
19086: Procedure mnuPrintFont1Click( Sender : TObject)
19087: procedure ModalStarted
19088: Procedure Modified
19089: Procedure ModifyAlias( Name : string; List : TStrings)
19090: Procedure ModifyDriver( Name : string; List : TStrings)
19091: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
19092: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
19093: Procedure Move( CurIndex, NewIndex : Integer)
```

```
19094: procedure Move(CurIndex, NewIndex: Integer);
19095: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
19096: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
19097: Procedure moveCube( o : TMyLabel)
19098: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
19099: procedure MoveTo(X, Y: Integer);
19100: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
19101: Procedure MovePoint(var x,y:Extended; const angle:Extended);
19102: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
19103: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
19104: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
19105: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
19106: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
19107: procedure New(P: PChar)
19108: procedure New1Click(Sender: TObject);
19109: procedure NewInstance1Click(Sender: TObject);
19110: Procedure NEXT
19111: Procedure NextMonth
19112: Procedure Noop
19113: Procedure NormalizePath( var APath : string)
19114: procedure ObjectBinaryToText(Input, Output: TStream)
19115: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19116: procedure ObjectResourceToText(Input, Output: TStream)
19117: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19118: procedure ObjectTextToBinary(Input, Output: TStream)
19119: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19120: procedure ObjectTextToResource(Input, Output: TStream)
19121: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19122: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
19123: Procedure Open( const UserID : WideString; const Password : WideString);
19124: Procedure Open;
19125: Procedure open1Click( Sender : TObject)
19126: Procedure OpenCdDrive
19127: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
19128: Procedure OpenCurrent
19129: Procedure OpenFile(vfilenamepath: string)
19130: Procedure OpenDirectory1Click( Sender : TObject)
19131: Procedure OpenDir(adir: string);
19132: Procedure OpenIndexFile( const IndexName : string)
19133: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
19134: Procedure OpenWriteBuffer( const AThreshhold : Integer)
19135: Procedure OptimizeMem
19136: Procedure Options1( AURL : string);
19137: Procedure OutputDebugString(lpOutputString : PChar)
19138: Procedure PackBuffer
19139: Procedure Paint
19140: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
19141: Procedure PaintToTBitmap( Target : TBitmap)
19142: Procedure PaletteChanged
19143: Procedure ParentBiDiModeChanged
19144: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
19145: Procedure PasteFromClipboard;
19146: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
19147: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
19148: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
19149: Procedure PError( Text : string)
19150: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
19151: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
19152: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
19153: procedure playmp3(mpath: string);
19154: Procedure PlayMP31Click( Sender : TObject)
19155: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
19156: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
19157: procedure PolyBezier(const Points: array of TPoint);
19158: procedure PolyBezierTo(const Points: array of TPoint);
19159: procedure Polygon(const Points: array of TPoint);
19160: procedure Polyline(const Points: array of TPoint);
19161: Procedure Pop
19162: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
19163: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
19164: Procedure POPUP( X, Y : INTEGER)
19165: Procedure PopupURL(URL : WideString);
19166: Procedure POST
19167: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
19168: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
19169: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
19170: Procedure PostUser( const Email, FirstName, LastName : WideString)
19171: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19172: procedure Pred(X: int64);
19173: Procedure Prepare
19174: Procedure PrepareStatement
19175: Procedure PreProcessXML( AList : TStrings)
19176: Procedure PreventDestruction
19177: Procedure Print( const Caption : string)
19178: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
19179: procedure printf(const format: String; const args: array of const);
19180: Procedure PrintList(Value: TStringList);
19181: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
```

```
19182: Procedure Printout1Click( Sender : TObject)
19183: Procedure ProcessHeaders
19184: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
19185: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
19186: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
19187: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
19188: Procedure ProcessMessagesOFF;  //application.processmessages
19189: Procedure ProcessMessagesON;
19190: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
19191: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
19192: Procedure Proclist Size is: 3797 /1415
19193: Procedure procMessClick( Sender : TObject)
19194: Procedure PSScriptCompile( Sender : TPSScript)
19195: Procedure PSScriptExecute( Sender : TPSScript)
19196: Procedure PSScriptLine( Sender : TObject)
19197: Procedure Push( ABoundary : string)
19198: procedure PushItem(AItem: Pointer)
19199: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
19200: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
19201: procedure PutLinuxLines(const Value: string)
19202: Procedure Quit
19203: Procedure RaiseConversionError( const AText : string);
19204: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
19205: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
19206: procedure RaiseException(Ex: TIFException; Param: String);
19207: Procedure RaiseExceptionForLastCmdResult;
19208: procedure RaiseLastException;
19209: procedure RaiseException2;
19210: Procedure RaiseException3(const Msg: string);
19211: Procedure RaiseExcept(const Msg: string);
19212: Procedure RaiseLastOSError
19213: Procedure RaiseLastWin32;
19214: procedure RaiseLastWin32Error)
19215: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
19216: Procedure RandomFillStream( Stream : TMemoryStream)
19217: procedure randomize;
19218: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
19219: Procedure RCS
19220: Procedure Read( Socket : TSocket)
19221: procedure Readln1(var ast: string);  //of inputquery
19222: Procedure ReadBlobData
19223: procedure ReadBuffer(Buffer:String;Count:LongInt)
19224: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
19225: Procedure ReadSection( const Section : string; Strings : TStrings)
19226: Procedure ReadSections( Strings : TStrings)
19227: Procedure ReadSections( Strings : TStrings);
19228: Procedure ReadSections1( const Section : string; Strings : TStrings);
19229: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
19230: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
19231: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
19232: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
19233: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
19234: Procedure Realign;
19235: procedure Rectangle(X1, Y1, X2, Y2: Integer);
19236: Procedure Rectangle1( const Rect : TRect);
19237: Procedure RectCopy( var Dest : TRect; const Source : TRect)
19238: Procedure RectFitToScreen( var R : TRect)
19239: Procedure RectGrow( var R : TRect; const Delta : Integer)
19240: Procedure RectGrowX( var R : TRect; const Delta : Integer)
19241: Procedure RectGrowY( var R : TRect; const Delta : Integer)
19242: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
19243: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
19244: Procedure RectNormalize( var R : TRect)
19245: //  TFileCallbackProcedure = procedure(filename:string);
19246: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
19247: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
19248: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
19249: Procedure Refresh;
19250: Procedure RefreshData( Options : TFetchOptions)
19251: Procedure REFRESHLOOKUPLIST
19252: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
19253: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
19254: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
19255: Procedure RegisterChanges( Value : TChangeLink)
19256: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
19257: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
19258: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
19259: Procedure ReInitialize( ADelay : Cardinal)
19260: procedure RELEASE
19261: Procedure Remove( const AByteCount : integer)
19262: Procedure REMOVE( FIELD : TFIELD)
19263: Procedure REMOVE( ITEM : TMENUITEM)
19264: Procedure REMOVE( POPUP : TPOPUPMENU)
19265: Procedure RemoveAllPasswords
19266: procedure RemoveComponent(AComponent:TComponent)
19267: Procedure RemoveDir( const ADirName : string)
19268: Procedure RemoveLambdaTransitions( var bChanged : boolean)
19269: Procedure REMOVEPARAM( VALUE : TPARAM)
19270: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
```

```
19271: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
19272: Procedure Rename( const ASourceFile, ADestFile : string)
19273: Procedure Rename( const FileName : string; Reload : Boolean)
19274: Procedure RenameTable( const NewTableName : string)
19275: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
19276: Procedure Replace1Click( Sender : TObject)
19277: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
19278: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
19279: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
19280: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
19281: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
19282: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
19283: Procedure Requery( Options : TExecuteOptions)
19284: Procedure Reset
19285: Procedure Reset1Click( Sender : TObject)
19286: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
19287: procedure ResourceExplore1Click(Sender: TObject);
19288: Procedure RestoreContents
19289: Procedure RestoreDefaults
19290: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
19291: Procedure RetrieveHeaders
19292: Procedure RevertRecord
19293: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
19294: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19295: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19296: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
19297: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
19298: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
19299: Procedure RleCompress2( Stream : TStream)
19300: Procedure RleDecompress2( Stream : TStream)
19301: Procedure RmDir(const S: string)
19302: Procedure Rollback
19303: Procedure Rollback( TransDesc : TTransactionDesc)
19304: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
19305: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
19306: Procedure RollbackTrans
19307: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
19308: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
19309: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
19310: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
19311: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
19312: Procedure S_EBox( const AText : string)
19313: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
19314: Procedure S_IBox( const AText : string)
19315: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
19316: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
19317: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
19318: Procedure SampleVarianceAndMean
19319: ( const X : TDynFloatArray; var Variance, Mean : Float)
19320: Procedure Save2Click( Sender : TObject)
19321: Procedure Saveas3Click( Sender : TObject)
19322: Procedure Savebefore1Click( Sender : TObject)
19323: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
19324: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19325: Procedure SaveConfigFile
19326: Procedure SaveOutput1Click( Sender : TObject)
19327: procedure SaveScreenshotClick(Sender: TObject);
19328: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
19329: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
19330: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
19331: Procedure SaveToFile( AFileName : string)
19332: Procedure SAVETOFILE( const FILENAME : String)
19333: Procedure SaveToFile( const FileName : WideString)
19334: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
19335: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19336: procedure SaveToFile(FileName: string);
19337: procedure SaveToFile(FileName:String)
19338: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
19339: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
19340: Procedure SaveToStream( S : TStream)
19341: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19342: Procedure SaveToStream( Stream : TStream)
19343: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
19344: procedure SaveToStream(Stream: TStream);
19345: procedure SaveToStream(Stream:TStream)
19346: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
19347: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
19348: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
19349: procedure Say(const sText: string)
19350: Procedure SBytecode1Click( Sender : TObject)
19351: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
19352: procedure ScriptExplorer1Click(Sender: TObject);
19353: Procedure Scroll( Distance : Integer)
19354: Procedure Scroll( DX, DY : Integer)
19355: procedure ScrollBy(DeltaX, DeltaY: Integer);
19356: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
19357: Procedure ScrollTabs( Delta : Integer)
19358: Procedure Search1Click( Sender : TObject)
```

```
19359: procedure SearchAndOpenDoc(vfilenamepath: string)
19360: procedure SearchAndOpenFile(vfilenamepath: string)
19361: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
19362: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19363: Procedure SearchNext1Click( Sender : TObject)
19364: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
19365: Procedure Select1( const Nodes : array of TTreeNode);
19366: Procedure Select2( Nodes : TList);
19367: Procedure SelectNext( Direction : Boolean)
19368: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
19369: Procedure SelfTestPEM  //unit uPSI_cPEM
19370: Procedure Send( AMsg : TIdMessage)
19371: //config forst in const MAILINIFILE = 'maildef.ini';
19372: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
19373: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19374: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19375: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
19376: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
19377: Procedure SendResponse
19378: Procedure SendStream( AStream : TStream)
19379: procedure SendMCICommand(Cmd: string);   !
19380: Procedure Set8087CW( NewCW : Word)
19381: Procedure SetAll( One, Two, Three, Four : Byte)
19382: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
19383: Procedure SetAppDispatcher( const ADispatcher : TComponent)
19384: procedure SetArrayLength;
19385: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
19386: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19387: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer);  //all init
19388: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
19389: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);');
19390: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);');
19391: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
19392: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
19393: Procedure SetAsHandle( Format : Word; Value : THandle)
19394: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
19395: procedure SetCaptureControl(Control: TControl);
19396: Procedure SetColumnAttributes
19397: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
19398: Procedure SetCustomHeader( const Name, Value : string)
19399: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
19400: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
19401: Procedure SetFocus
19402: procedure SetFocus; virtual;
19403: Procedure SetInitialState
19404: Procedure SetKey
19405: procedure SetLastError(ErrorCode: Integer)
19406: procedure SetLength;
19407: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
19408: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
19409: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
19410: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
19411: Procedure SetParams1( UpdateKind : TUpdateKind);
19412: Procedure SetPassword( const Password : string)
19413: Procedure SetPointer( Ptr : Pointer; Size : Longint)
19414: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
19415: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
19416: Procedure SetProvider( Provider : TComponent)
19417: Procedure SetProxy( const Proxy : string)
19418: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
19419: Procedure SetRange( const StartValues, EndValues : array of const)
19420: Procedure SetRangeEnd
19421: Procedure SetRate( const aPercent, aYear : integer)
19422: procedure SetRate(const aPercent, aYear: integer)
19423: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19424: Procedure SetSafeCallExceptionMsg( Msg : String)
19425: procedure SETSELTEXTBUF(BUFFER:PCHAR)
19426: Procedure SetSize( AWidth, AHeight : Integer)
19427: procedure SetSize(NewSize:LongInt)
19428: procedure SetString(var s: string; buffer: PChar; len: Integer)
19429: Procedure SetStrings( List : TStrings)
19430: Procedure SetText( Text : PwideChar)
19431: procedure SetText(Text: PChar);
19432: Procedure SetTextBuf( Buffer : PChar)
19433: procedure SETTEXTBUF(BUFFER:PCHAR)
19434: Procedure SetTick( Value : Integer)
19435: Procedure SetTimeout( ATimeOut : Integer)
19436: Procedure SetTraceEvent( Event : TDBXTraceEvent)
19437: Procedure SetUserName( const UserName : string)
19438: Procedure SetWallpaper( Path : string);
19439: procedure ShellStyle1Click(Sender: TObject);
19440: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
19441: Procedure ShowFileProperties( const FileName : string)
19442: Procedure ShowInclude1Click( Sender : TObject)
19443: Procedure ShowInterfaces1Click( Sender : TObject)
19444: Procedure ShowLastException1Click( Sender : TObject)
19445: Procedure ShowMessage( const Msg : string)
19446: Procedure ShowMessageBig(const aText : string);
```

```
19447: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
19448: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
19449: Procedure MsgBig(const aText : string);        //alias
19450: procedure showmessage(mytext: string);
19451: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
19452: procedure ShowMessageFmt(const Msg: string; Params: array of const))
19453: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
19454: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
19455: Procedure ShowSearchDialog( const Directory : string)
19456: Procedure ShowSpecChars1Click( Sender : TObject)
19457: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
19458: Procedure ShredFile( const FileName : string; Times : Integer)
19459: procedure Shuffle(vQ: TStringList);
19460: Procedure ShuffleList( var List : array of Integer; Count : Integer)
19461: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
19462: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
19463: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
19464: Procedure Site( const ACommand : string)
19465: Procedure SkipEOL
19466: Procedure Sleep( ATime : cardinal)
19467: Procedure Sleep( milliseconds : Cardinal)
19468: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
19469: Procedure Slinenumbers1Click( Sender : TObject)
19470: Procedure Sort
19471: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
19472: procedure SoundAlarm;  //beep seq
19473: procedure Speak(const sText: string)   //async like voice
19474: procedure Speak2(const sText: string)  //sync
19475: procedure Split(Str: string;  SubStr: string; List: TStrings);
19476: Procedure SplitNameValue( const Line : string; var Name, Value : string)
19477: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
19478: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
19479: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
19480: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
19481: procedure SQLSyntax1Click(Sender: TObject);
19482: Procedure SRand( Seed : RNG_IntType)
19483: Procedure Start
19484: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
19485: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
19486: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
19487: Procedure StartTransaction( TransDesc : TTransactionDesc)
19488: Procedure Status( var AStatusList : TStringList)
19489: Procedure StatusBar1DblClick( Sender : TObject)
19490: Procedure StepInto1Click( Sender : TObject)
19491: Procedure StepIt
19492: Procedure StepOut1Click( Sender : TObject)
19493: Procedure Stop
19494: procedure stopmp3;
19495: procedure StartWeb(aurl: string);
19496: Procedure Str(aint: integer; astr: string); //of system
19497: Procedure StrDispose( Str : PChar)
19498: procedure StrDispose(Str: PChar)
19499: Procedure StrReplace(var Str: String; Old, New: String);
19500: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
19501: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
19502: Procedure StringToBytes( Value : String; Bytes : array of byte)
19503: procedure StrSet(c : Char; I : Integer; var s : String);
19504: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19505: Procedure StructureMount( APath : String)
19506: procedure STYLECHANGED(SENDER:TOBJECT)
19507: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
19508: procedure Succ(X: int64);
19509: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
19510: procedure SwapChar(var X,Y: char);  //swapX follows
19511: Procedure SwapFloats( var X, Y : Float)
19512: procedure SwapGrid(grd: TStringGrid);
19513: Procedure SwapOrd( var I, J : Byte);
19514: Procedure SwapOrd( var X, Y : Integer)
19515: Procedure SwapOrd1( var I, J : Shortint);
19516: Procedure SwapOrd2( var I, J : Smallint);
19517: Procedure SwapOrd3( var I, J : Word);
19518: Procedure SwapOrd4( var I, J : Integer);
19519: Procedure SwapOrd5( var I, J : Cardinal);
19520: Procedure SwapOrd6( var I, J : Int64);
19521: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
19522: Procedure Synchronize1( Method : TMethod);
19523: procedure SyntaxCheck1Click(Sender: TObject);
19524: Procedure SysFreeString(const S: WideString); stdcall;
19525: Procedure TakeOver( Other : TLinearBitmap)
19526: Procedure Talkln(const sText: string)  //async voice
19527: procedure tbtn6resClick(Sender: TObject);
19528: Procedure tbtnUseCaseClick( Sender : TObject)
19529: procedure TerminalStyle1Click(Sender: TObject);
19530: Procedure Terminate
19531: Procedure texSyntax1Click( Sender : TObject)
19532: procedure TextOut(X, Y: Integer; Text: string);
19533: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
19534: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
19535: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
```

```
19536: Procedure TextStart
19537: procedure TILE
19538: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
19539: Procedure TitleClick( Column : TColumn)
19540: Procedure ToDo
19541: procedure toolbtnTutorialClick(Sender: TObject);
19542: Procedure Trace1( AURL : string; const AResponseContent : TStream);
19543: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
19544: Procedure Truncate
19545: procedure Tutorial101Click(Sender: TObject);
19546: procedure Tutorial10Statistics1Click(Sender: TObject);
19547: procedure Tutorial11Forms1Click(Sender: TObject);
19548: procedure Tutorial12SQL1Click(Sender: TObject);
19549: Procedure tutorial1Click( Sender : TObject)
19550: Procedure tutorial21Click( Sender : TObject)
19551: Procedure tutorial31Click( Sender : TObject)
19552: Procedure tutorial4Click( Sender : TObject)
19553: Procedure Tutorial5Click( Sender : TObject)
19554: procedure Tutorial6Click(Sender: TObject);
19555: procedure Tutorial91Click(Sender: TObject);
19556: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19557: procedure UniqueString(var str: AnsiString)
19558: procedure UnloadLoadPackage(Module: HMODULE)
19559: Procedure Unlock
19560: Procedure UNMERGE( MENU : TMAINMENU)
19561: Procedure UnRegisterChanges( Value : TChangeLink)
19562: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19563: Procedure UnregisterConversionType( const AType : TConvType)
19564: Procedure UnRegisterProvider( Prov : TCustomProvider)
19565: Procedure UPDATE
19566: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19567: Procedure UPDATECURSORPOS
19568: Procedure UpdateFile
19569: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19570: Procedure UpdateResponse( AResponse : TWebResponse)
19571: Procedure UpdateScrollBar
19572: Procedure UpdateView1Click( Sender : TObject)
19573: procedure Val(const s: string; var n, z: Integer)
19574: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19575: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
19576: Procedure VariantAdd( const src : Variant; var dst : Variant)
19577: Procedure VariantAnd( const src : Variant; var dst : Variant)
19578: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19579: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19580: Procedure VariantClear( var V : Variant)
19581: Procedure VariantCpy( const src : Variant; var dst : Variant)
19582: Procedure VariantDiv( const src : Variant; var dst : Variant)
19583: Procedure VariantMod( const src : Variant; var dst : Variant)
19584: Procedure VariantMul( const src : Variant; var dst : Variant)
19585: Procedure VariantOr( const src : Variant; var dst : Variant)
19586: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
19587: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
19588: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
19589: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
19590: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
19591: Procedure VariantShl( const src : Variant; var dst : Variant)
19592: Procedure VariantShr( const src : Variant; var dst : Variant)
19593: Procedure VariantSub( const src : Variant; var dst : Variant)
19594: Procedure VariantXor( const src : Variant; var dst : Variant)
19595: Procedure VarCastError;
19596: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19597: Procedure VarInvalidOp
19598: Procedure VarInvalidNullOp
19599: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19600: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19601: Procedure VarArrayCreateError
19602: Procedure VarResultCheck( AResult : HRESULT);
19603: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19604: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19605: Function VarTypeAsText( const AType : TVarType) : string
19606: procedure Voice(const sText: string) //async
19607: procedure Voice2(const sText: string) //sync
19608: Procedure WaitMiliSeconds( AMSec : word)
19609: Procedure WaitMS( AMSec : word)');
19610: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
19611: Procedure WideAppend( var dst : WideString; const src : WideString)
19612: Procedure WideAssign( var dst : WideString; var src : WideString)
19613: Procedure WideDelete( var dst : WideString; index, count : Integer)
19614: Procedure WideFree( var s : WideString)
19615: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19616: Procedure WideFromPChar( var dst : WideString; src : PChar)
19617: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19618: Procedure WideSetLength( var dst : WideString; len : Integer)
19619: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
19620: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
19621: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19622: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19623: Procedure HttpGet(const Url: string; Stream:TStream);
19624: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
```

```
19625: Procedure WordWrap1Click( Sender : TObject)
19626: Procedure Write( const AOut : string)
19627: Procedure Write( Socket : TSocket)
19628: procedure Write(S: string);
19629: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19630: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19631: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19632: procedure WriteBuffer(Buffer:String;Count:LongInt)
19633: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19634: Procedure WriteChar( AValue : Char)
19635: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19636: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19637: Procedure WriteFloat( const Section, Name : string; Value : Double)
19638: Procedure WriteHeader( AHeader : TStrings)
19639: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19640: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19641: Procedure WriteLn( const AOut : string)
19642: procedure Writeln(s: string);
19643: Procedure WriteLog( const FileName, LogLine : string)
19644: Procedure WriteRFCReply( AReply : TIdRFCReply)
19645: Procedure WriteRFCStrings( AStrings : TStrings)
19646: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19647: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
19648: Procedure WriteString( const Section, Ident, Value : String)
19649: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19650: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19651: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19652: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19653: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19654: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
19655: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19656: procedure XMLSyntax1Click(Sender: TObject);
19657: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19658: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19659: Procedure ZeroFillStream( Stream : TMemoryStream)
19660: procedure XMLSyntax1Click(Sender: TObject);
19661: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19662: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19663: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19664: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19665: procedure(Sender, Source: TObject;X, Y: Integer)
19666: procedure(Sender, Target: TObject; X, Y: Integer)
19667: procedure(Sender: TObject; ASection, AWidth: Integer)
19668: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
19669: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
19670: procedure(Sender: TObject; var Action: TCloseAction)
19671: procedure(Sender: TObject; var CanClose: Boolean)
19672: procedure(Sender: TObject; var Key: Char);
19673: ProcedureName ProcedureNames ProcedureParametersCursor @
19674:
19675: *************Now Constructors constructor *************
19676:  Size is: 1248 1115 996 628 550 544 501 459 (381)
19677:  Attach( VersionInfoData : Pointer; Size : Integer)
19678:  constructor Create( ABuckets : TBucketListSizes)
19679:  Create( ACallBackWnd : HWND)
19680:  Create( AClient : TCustomTaskDialog)
19681:  Create( AClient : TIdTelnet)
19682:  Create( ACollection : TCollection)
19683:  Create( ACollection : TFavoriteLinkItems)
19684:  Create( ACollection : TTaskDialogButtons)
19685:  Create( AConnection : TIdCustomHTTP)
19686:  Create( ACreateSuspended : Boolean)
19687:  Create( ADataSet : TCustomSQLDataSet)
19688:  CREATE( ADATASET : TDATASET)
19689:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19690:  Create( AGrid : TCustomDBGrid)
19691:  Create( AGrid : TStringGrid; AIndex : Longint)
19692:  Create( AHTTP : TIdCustomHTTP)
19693:  Create( AListItems : TListItems)
19694:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19695:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19696:  Create( AOwner : TCommonCalendar)
19697:  Create( AOwner : TComponent)
19698:  CREATE( AOWNER : TCOMPONENT)
19699:  Create( AOwner : TCustomListView)
19700:  Create( AOwner : TCustomOutline)
19701:  Create( AOwner : TCustomRichEdit)
19702:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19703:  Create( AOwner : TCustomTreeView)
19704:  Create( AOwner : TIdUserManager)
19705:  Create( AOwner : TListItems)
19706:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19707:  CREATE( AOWNER : TPERSISTENT)
19708:  Create( AOwner : TPersistent)
19709:  Create( AOwner : TTable)
19710:  Create( AOwner : TTreeNodes)
19711:  Create( AOwner : TWinControl; const ClassName : string)
19712:  Create( AParent : TIdCustomHTTP)
19713:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
```

```
19714:  Create( AProvider : TBaseProvider)
19715:  Create( AProvider : TCustomProvider);
19716:  Create( AProvider : TDataSetProvider)
19717:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19718:  Create( ASocket : TSocket)
19719:  Create( AStrings : TWideStrings)
19720:  Create( AToolBar : TToolBar)
19721:  Create( ATreeNodes : TTreeNodes)
19722:  Create( Autofill : boolean)
19723:  Create( AWebPageInfo : TAbstractWebPageInfo)
19724:  Create( AWebRequest : TWebRequest)
19725:  Create( Collection : TCollection)
19726:  Create( Collection : TIdMessageParts; ABody : TStrings)
19727:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
19728:  Create( Column : TColumn)
19729:  Create( const AConvFamily : TConvFamily; const ADescription : string)
19730:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
19731:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
19732:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
19733:  Create( const ATabSet : TTabSet)
19734:  Create( const Compensate : Boolean)
19735:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
19736:  Create( const FileName : string)
19737:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
19738:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
19739:  Create( const MaskValue : string)
19740:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19741:  Create( const Prefix : string)
19742:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
19743:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19744:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19745:  Create( CoolBar : TCoolBar)
19746:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
19747:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
19748:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
19749:  Create( DBCtrlGrid : TDBCtrlGrid)
19750:  Create( DSTableProducer : TDSTableProducer)
19751:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
19752:  Create( ErrorCode : DBIResult)
19753:  Create( Field : TBlobField; Mode : TBlobStreamMode)
19754:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
19755:  Create( HeaderControl : TCustomHeaderControl)
19756:  Create( HTTPRequest : TWebRequest)
19757:  Create( iStart : integer; sText : string)
19758:  Create( iValue : Integer)
19759:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19760:  Create( MciErrNo : MCIERROR; const Msg : string)
19761:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19762:  Create( Message : string; ErrorCode : DBResult)
19763:  Create( Msg : string)
19764:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19765:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19766:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19767:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19768:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
19769:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19770:  Create( Owner : TCustomComboBoxEx)
19771:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19772:  Create( Owner : TPersistent)
19773:  Create( Params : TStrings)
19774:  Create( Size : Cardinal)
19775:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19776:  Create( StatusBar : TCustomStatusBar)
19777:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
19778:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19779:  Create(AHandle:Integer)
19780:  Create(AOwner: TComponent); virtual;
19781:  Create(const AURI : string)
19782:  Create(FileName:String;Mode:Word)
19783:  Create(Instance:THandle;ResName:String;ResType:PChar)
19784:  Create(Stream : TStream)
19785:  Create1( ADataset : TDataset);
19786:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
19787:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19788:  Create2( Other : TObject);
19789:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19790:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19791:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
19792:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19793:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19794:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19795:  CreateRes( Ident : Integer);
19796:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
19797:  CreateRes( ResStringRec : PResStringRec);
```

```
19798:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
19799:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19800:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19801:  CreateSize( AWidth, AHeight : Integer)
19802:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19803:
19804: ------------------------------------------------------------------------------
19805: unit uPSI_MathMax;
19806: ------------------------------------------------------------------------------
19807:  CONSTS
19808:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
19809:   Cbrt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
19810:   Cbrt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
19811:   Cbrt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
19812:   Cbrt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
19813:   CbrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
19814:   Catalan: Float   = 0.91596559417721901505460351493 24;  // Catalan constant
19815:   PiJ:   Float       = 3.14159265358979323846264338327 95;  // PI
19816:   PI:  Extended =  3.14159265358979323846264338327 95);
19817:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
19818:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
19819:   PiOn4: Float     = 0.78539816339744830961566084581 988; // PI / 4
19820:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
19821:   Sqrt3: Float     = 1.73205080756887729352744634150 59;  // Sqrt(3)
19822:   Sqrt5: Float     = 2.23606797749978969640917366873 13;  // Sqrt(5)
19823:   Sqrt10: Float    = 3.16227766016837933199889354443 27;  // Sqrt(10)
19824:   SqrtPi: Float    = 1.77245385090551602729816748334 11;  // Sqrt(PI)
19825:   Sqrt2Pi: Float   = 2.50662827463100050241576528481 1;   // Sqrt(2 * PI)
19826:   TwoPi: Float     = 6.28318530717958647692528676655 9;   // 2 * PI
19827:   ThreePi: Float   = 9.42477796076937971538793014983 85;  // 3 * PI
19828:   Ln2: Float       = 0.69314718055994530941723212145 818; // Ln(2)
19829:   Ln10: Float      = 2.30258509299404568401799145468 44;  // Ln(10)
19830:   LnPi: Float      = 1.14472988584940017414342735135 31;  // Ln(PI)
19831:   Log2J: Float     = 0.30102999566398119521373889472 449; // Log10(2)
19832:   Log3: Float      = 0.47712125471966243729502790325 512; // Log10(3)
19833:   LogPi: Float     = 0.49714987269413385435126828829 09;  // Log10(PI)
19834:   LogE: Float      = 0.43429448190325182765112891891 661; // Log10(E)
19835:   E: Float         = 2.71828182845904523536028747135 27;  // Natural constant
19836:   hLn2Pi: Float    = 0.91893853320467274178032973640 562; // Ln(2*PI)/2
19837:   inv2Pi: Float    = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
19838:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
19839:   GoldenMean: Float   = 1.61803398874989484820458683436563 8;  // GoldenMean
19840:   EulerMascheroni: Float = 0.57721566490153286060651209008 24;  // Euler GAMMA
19841:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
19842:   StDelta         : Extended = 0.00001;  {delta for difference equations}
19843:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
19844:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
19845:
19846: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
19847: begin
19848:   MetersPerInch = 0.0254; // [1]
19849:   MetersPerFoot = MetersPerInch * 12;
19850:   MetersPerYard = MetersPerFoot * 3;
19851:   MetersPerMile = MetersPerFoot * 5280;
19852:   MetersPerNauticalMiles = 1852;
19853:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
19854:   MetersPerLightSecond = 2.99792458E8; // [5]
19855:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
19856:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
19857:   MetersPerCubit = 0.4572; // [6][7]
19858:   MetersPerFathom = MetersPerFoot * 6;
19859:   MetersPerFurlong = MetersPerYard * 220;
19860:   MetersPerHand = MetersPerInch * 4;
19861:   MetersPerPace = MetersPerInch * 30;
19862:   MetersPerRod = MetersPerFoot * 16.5;
19863:   MetersPerChain = MetersPerRod * 4;
19864:   MetersPerLink = MetersPerChain / 100;
19865:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
19866:   MetersPerPica = MetersPerPoint * 12;
19867:
19868:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
19869:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
19870:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
19871:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
19872:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
19873:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
19874:
19875:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
19876:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
19877:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
19878:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
19879:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
19880:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
19881:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
19882:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
19883:
19884:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
19885:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
19886:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
```

```
19887:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
19888:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
19889:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
19890:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
19891:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
19892:
19893:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
19894:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
19895:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
19896:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
19897:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
19898:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
19899:
19900:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
19901:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
19902:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
19903:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
19904:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
19905:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
19906:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
19907:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
19908:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
19909:
19910:   GramsPerPound = 453.59237; // [1][7]
19911:   GramsPerDrams = GramsPerPound / 256;
19912:   GramsPerGrains = GramsPerPound / 7000;
19913:   GramsPerTons = GramsPerPound * 2000;
19914:   GramsPerLongTons = GramsPerPound * 2240;
19915:   GramsPerOunces = GramsPerPound / 16;
19916:   GramsPerStones = GramsPerPound * 14;
19917:
19918:   MaxAngle 9223372036854775808.0;
19919:   MaxTanH 5678.26170314707197474596553898 54);
19920:   MaxFactorial( 1754);
19921:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
19922:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
19923:   MaxTanH( 354.89135644669199842162284618659);
19924:   MaxFactorial'LongInt'( 170);
19925:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
19926:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
19927:   MaxTanH( 44.361419555836499802702855773323);
19928:   MaxFactorial'LongInt'( 33);
19929:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
19930:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
19931:   PiExt( 3.14159265358979323846264338327 95);
19932:   RatioDegToRad( PiExt / 180.0);
19933:   RatioGradToRad( PiExt / 200.0);
19934:   RatioDegToGrad( 200.0 / 180.0);
19935:   RatioGradToDeg( 180.0 / 200.0);
19936:   Crc16PolynomCCITT'LongWord $1021);
19937:  Crc16PolynomIBM'LongWord $8005);
19938:  Crc16Bits'LongInt'( 16);
19939:  Crc16Bytes'LongInt'( 2);
19940:  Crc16HighBit'LongWord $8000);
19941:  NotCrc16HighBit','LongWord $7FFF);
19942:   Crc32PolynomIEEE','LongWord $04C11DB7);
19943:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
19944:  Crc32Koopman','LongWord $741B8CD7);
19945:  Crc32Bits','LongInt'( 32);
19946:  Crc32Bytes','LongInt'( 4);
19947:  Crc32HighBit','LongWord $80000000);
19948:  NotCrc32HighBit','LongWord $7FFFFFFF);
19949:
19950:   MinByte       = Low(Byte);
19951:   MaxByte       = High(Byte);
19952:   MinWord       = Low(Word);
19953:   MaxWord       = High(Word);
19954:   MinShortInt   = Low(ShortInt);
19955:   MaxShortInt   = High(ShortInt);
19956:   MinSmallInt   = Low(SmallInt);
19957:   MaxSmallInt   = High(SmallInt);
19958:   MinLongWord   = LongWord(Low(LongWord));
19959:   MaxLongWord   = LongWord(High(LongWord));
19960:   MinLongInt    = LongInt(Low(LongInt));
19961:   MaxLongInt    = LongInt(High(LongInt));
19962:   MinInt64      = Int64(Low(Int64));
19963:   MaxInt64      = Int64(High(Int64));
19964:   MinInteger    = Integer(Low(Integer));
19965:   MaxInteger    = Integer(High(Integer));
19966:   MinCardinal   = Cardinal(Low(Cardinal));
19967:   MaxCardinal   = Cardinal(High(Cardinal));
19968:   MinNativeUInt = NativeUInt(Low(NativeUInt));
19969:   MaxNativeUInt = NativeUInt(High(NativeUInt));
19970:   MinNativeInt  = NativeInt(Low(NativeInt));
19971:   MaxNativeInt  = NativeInt(High(NativeInt));
19972:  Function CosH( const Z : Float) : Float;
19973:  Function SinH( const Z : Float) : Float;
19974:  Function TanH( const Z : Float) : Float;
19975:
```

```
19976:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
19977:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
19978:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
19979:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
19980:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
19981:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
19982:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
19983:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
19984:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
19985:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
19986:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
19987:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
19988:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
19989:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
19990:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
19991:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
19992:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
19993:   MaxLog = 709.7827128933840;
19994:   MinLog = -708.3964185322641;
19995:   MaxFac = 170;
19996:   MaxGam = 171.624376956302;
19997:   MaxLgm = 2.556348E+305;
19998:   SingleCompareDelta   = 1.0E-34;
19999:   DoubleCompareDelta   = 1.0E-280;
20000:   {$IFDEF CLR}
20001:   ExtendedCompareDelta = DoubleCompareDelta;
20002:   {$ELSE}
20003:   ExtendedCompareDelta = 1.0E-4400;
20004:   {$ENDIF}
20005:   Bytes1KB  = 1024;
20006:   Bytes1MB  = 1024 * Bytes1KB;
20007:   Bytes1GB  = 1024 * Bytes1MB;
20008:   Bytes64KB = 64 * Bytes1KB;
20009:   Bytes64MB = 64 * Bytes1MB;
20010:   Bytes2GB  = 2 * LongWord(Bytes1GB);
20011:     clBlack32', $FF000000 ));
20012:     clDimGray32', $FF3F3F3F ));
20013:     clGray32', $FF7F7F7F ));
20014:     clLightGray32', $FFBFBFBF ));
20015:     clWhite32', $FFFFFFFF ));
20016:     clMaroon32', $FF7F0000 ));
20017:     clGreen32', $FF007F00 ));
20018:     clOlive32', $FF7F7F00 ));
20019:     clNavy32', $FF00007F ));
20020:     clPurple32', $FF7F007F ));
20021:     clTeal32', $FF007F7F ));
20022:     clRed32', $FFFF0000 ));
20023:     clLime32', $FF00FF00 ));
20024:     clYellow32', $FFFFFF00 ));
20025:     clBlue32', $FF0000FF ));
20026:     clFuchsia32', $FFFF00FF ));
20027:     clAqua32', $FF00FFFF ));
20028:     clAliceBlue32', $FFF0F8FF ));
20029:     clAntiqueWhite32', $FFFAEBD7 ));
20030:     clAquamarine32', $FF7FFFD4 ));
20031:     clAzure32', $FFF0FFFF ));
20032:     clBeige32', $FFF5F5DC ));
20033:     clBisque32', $FFFFE4C4 ));
20034:     clBlancheDalmond32', $FFFFEBCD ));
20035:     clBlueViolet32', $FF8A2BE2 ));
20036:     clBrown32', $FFA52A2A ));
20037:     clBurlyWood32', $FFDEB887 ));
20038:     clCadetblue32', $FF5F9EA0 ));
20039:     clChartReuse32', $FF7FFF00 ));
20040:     clChocolate32', $FFD2691E ));
20041:     clCoral32', $FFFF7F50 ));
20042:     clCornFlowerBlue32', $FF6495ED ));
20043:     clCornSilk32', $FFFFF8DC ));
20044:     clCrimson32', $FFDC143C ));
20045:     clDarkBlue32', $FF00008B ));
20046:     clDarkCyan32', $FF008B8B ));
20047:     clDarkGoldenRod32', $FFB8860B ));
20048:     clDarkGray32', $FFA9A9A9 ));
20049:     clDarkGreen32', $FF006400 ));
20050:     clDarkGrey32', $FFA9A9A9 ));
20051:     clDarkKhaki32', $FFBDB76B ));
20052:     clDarkMagenta32', $FF8B008B ));
20053:     clDarkOliveGreen32', $FF556B2F ));
20054:     clDarkOrange32', $FFFF8C00 ));
20055:     clDarkOrchid32', $FF9932CC ));
20056:     clDarkRed32', $FF8B0000 ));
20057:     clDarkSalmon32', $FFE9967A ));
20058:     clDarkSeaGreen32', $FF8FBC8F ));
20059:     clDarkSlateBlue32', $FF483D8B ));
20060:     clDarkSlateGray32', $FF2F4F4F ));
20061:     clDarkSlateGrey32', $FF2F4F4F ));
20062:     clDarkTurquoise32', $FF00CED1 ));
20063:     clDarkViolet32', $FF9400D3 ));
20064:     clDeepPink32', $FFFF1493 ));
```

```
20065:     clDeepSkyBlue32', $FF00BFFF ));
20066:     clDodgerBlue32', $FF1E90FF ));
20067:     clFireBrick32', $FFB22222 ));
20068:     clFloralWhite32', $FFFFFAF0 ));
20069:     clGainsBoro32', $FFDCDCDC ));
20070:     clGhostWhite32', $FFF8F8FF ));
20071:     clGold32', $FFFFD700 ));
20072:     clGoldenRod32', $FFDAA520 ));
20073:     clGreenYellow32', $FFADFF2F ));
20074:     clGrey32', $FF808080 ));
20075:     clHoneyDew32', $FFF0FFF0 ));
20076:     clHotPink32', $FFFF69B4 ));
20077:     clIndianRed32', $FFCD5C5C ));
20078:     clIndigo32', $FF4B0082 ));
20079:     clIvory32', $FFFFFFF0 ));
20080:     clKhaki32', $FFF0E68C ));
20081:     clLavender32', $FFE6E6FA ));
20082:     clLavenderBlush32', $FFFFF0F5 ));
20083:     clLawnGreen32', $FF7CFC00 ));
20084:     clLemonChiffon32', $FFFFFACD ));
20085:     clLightBlue32', $FFADD8E6 ));
20086:     clLightCoral32', $FFF08080 ));
20087:     clLightCyan32', $FFE0FFFF ));
20088:     clLightGoldenRodYellow32', $FFFAFAD2 ));
20089:     clLightGreen32', $FF90EE90 ));
20090:     clLightGrey32', $FFD3D3D3 ));
20091:     clLightPink32', $FFFFB6C1 ));
20092:     clLightSalmon32', $FFFFA07A ));
20093:     clLightSeagreen32', $FF20B2AA ));
20094:     clLightSkyblue32', $FF87CEFA ));
20095:     clLightSlategray32', $FF778899 ));
20096:     clLightSlategrey32', $FF778899 ));
20097:     clLightSteelblue32', $FFB0C4DE ));
20098:     clLightYellow32', $FFFFFFE0 ));
20099:     clLtGray32', $FFC0C0C0 ));
20100:     clMedGray32', $FFA0A0A4 ));
20101:     clDkGray32', $FF808080 ));
20102:     clMoneyGreen32', $FFC0DCC0 ));
20103:     clLegacySkyBlue32', $FFA6CAF0 ));
20104:     clCream32', $FFFFFBF0 ));
20105:     clLimeGreen32', $FF32CD32 ));
20106:     clLinen32', $FFFAF0E6 ));
20107:     clMediumAquamarine32', $FF66CDAA ));
20108:     clMediumBlue32', $FF0000CD ));
20109:     clMediumOrchid32', $FFBA55D3 ));
20110:     clMediumPurple32', $FF9370DB ));
20111:     clMediumSeaGreen32', $FF3CB371 ));
20112:     clMediumSlateBlue32', $FF7B68EE ));
20113:     clMediumSpringGreen32', $FF00FA9A ));
20114:     clMediumTurquoise32', $FF48D1CC ));
20115:     clMediumVioletRed32', $FFC71585 ));
20116:     clMidnightBlue32', $FF191970 ));
20117:     clMintCream32', $FFF5FFFA ));
20118:     clMistyRose32', $FFFFE4E1 ));
20119:     clMoccasin32', $FFFFE4B5 ));
20120:     clNavajoWhite32', $FFFFDEAD ));
20121:     clOldLace32', $FFFDF5E6 ));
20122:     clOliveDrab32', $FF6B8E23 ));
20123:     clOrange32', $FFFFA500 ));
20124:     clOrangeRed32', $FFFF4500 ));
20125:     clOrchid32', $FFDA70D6 ));
20126:     clPaleGoldenRod32', $FFEEE8AA ));
20127:     clPaleGreen32', $FF98FB98 ));
20128:     clPaleTurquoise32', $FFAFEEEE ));
20129:     clPaleVioletred32', $FFDB7093 ));
20130:     clPapayaWhip32', $FFFFEFD5 ));
20131:     clPeachPuff32', $FFFFDAB9 ));
20132:     clPeru32', $FFCD853F ));
20133:     clPlum32', $FFDDA0DD ));
20134:     clPowderBlue32', $FFB0E0E6 ));
20135:     clRosyBrown32', $FFBC8F8F ));
20136:     clRoyalBlue32', $FF4169E1 ));
20137:     clSaddleBrown32', $FF8B4513 ));
20138:     clSalmon32', $FFFA8072 ));
20139:     clSandyBrown32', $FFF4A460 ));
20140:     clSeaGreen32', $FF2E8B57 ));
20141:     clSeaShell32', $FFFFF5EE ));
20142:     clSienna32', $FFA0522D ));
20143:     clSilver32', $FFC0C0C0 ));
20144:     clSkyblue32', $FF87CEEB ));
20145:     clSlateBlue32', $FF6A5ACD ));
20146:     clSlateGray32', $FF708090 ));
20147:     clSlateGrey32', $FF708090 ));
20148:     clSnow32', $FFFFFAFA ));
20149:     clSpringgreen32', $FF00FF7F ));
20150:     clSteelblue32', $FF4682B4 ));
20151:     clTan32', $FFD2B48C ));
20152:     clThistle32', $FFD8BFD8 ));
20153:     clTomato32', $FFFF6347 ));
```

```
20154:     clTurquoise32', $FF40E0D0 ));
20155:     clViolet32', $FFEE82EE ));
20156:     clWheat32', $FFF5DEB3 ));
20157:     clWhitesmoke32', $FFF5F5F5 ));
20158:     clYellowgreen32', $FF9ACD32 ));
20159:     clTrWhite32', $7FFFFFFF ));
20160:     clTrBlack32', $7F000000 ));
20161:     clTrRed32', $7FFF0000 ));
20162:     clTrGreen32', $7F00FF00 ));
20163:     clTrBlue32', $7F0000FF ));
20164:   // Fixed point math constants
20165:   FixedOne = $10000;  FixedHalf = $7FFF;
20166:   FixedPI  = Round(PI * FixedOne);
20167:   FixedToFloat = 1/FixedOne;
20168:
20169:  Special Types
20170: ************************************************
20171:   type Complex = record
20172:     X, Y : Float;
20173:   end;
20174:   type TVector     = array of Float;
20175:   TIntVector  = array of Integer;
20176:   TCompVector = array of Complex;
20177:   TBoolVector = array of Boolean;
20178:   TStrVector  = array of String;
20179:   TMatrix     = array of TVector;
20180:   TIntMatrix  = array of TIntVector;
20181:   TCompMatrix = array of TCompVector;
20182:   TBoolMatrix = array of TBoolVector;
20183:   TStrMatrix  = array of TStrVector;
20184:   TByteArray = array[0..32767] of byte; !
20185:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
20186:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
20187:   T2StringArray = array of array of string;
20188:   T2IntegerArray = array of array of integer;
20189:   AddTypeS('INT_PTR', 'Integer
20190:   AddTypeS('LONG_PTR', 'Integer
20191:   AddTypeS('UINT_PTR', 'Cardinal
20192:   AddTypeS('ULONG_PTR', 'Cardinal
20193:   AddTypeS('DWORD_PTR', 'ULONG_PTR
20194:   TIntegerDynArray', 'array of Integer
20195:   TCardinalDynArray', 'array of Cardinal
20196:   TWordDynArray', 'array of Word
20197:   TSmallIntDynArray', 'array of SmallInt
20198:   TByteDynArray', 'array of Byte
20199:   TShortIntDynArray', 'array of ShortInt
20200:   TInt64DynArray', 'array of Int64
20201:   TLongWordDynArray', 'array of LongWord
20202:   TSingleDynArray', 'array of Single
20203:   TDoubleDynArray', 'array of Double
20204:   TBooleanDynArray', 'array of Boolean
20205:   TStringDynArray', 'array of string
20206:   TWideStringDynArray', 'array of WideString
20207:   TDynByteArray     = array of Byte;
20208:   TDynShortintArray = array of Shortint;
20209:   TDynSmallintArray = array of Smallint;
20210:   TDynWordArray     = array of Word;
20211:   TDynIntegerArray  = array of Integer;
20212:   TDynLongintArray  = array of Longint;
20213:   TDynCardinalArray = array of Cardinal;
20214:   TDynInt64Array    = array of Int64;
20215:   TDynExtendedArray = array of Extended;
20216:   TDynDoubleArray   = array of Double;
20217:   TDynSingleArray   = array of Single;
20218:   TDynFloatArray    = array of Float;
20219:   TDynPointerArray  = array of Pointer;
20220:   TDynStringArray   = array of string;
20221:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
20222:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
20223:   TSynSearchOptions = set of TSynSearchOption;
20224:
20225:
20226: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
20227: ----------------------------------------------------------------------------
20228: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
20229: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
20230: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
20231: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
20232: function CheckStringSum(vstring: string): integer;
20233: function HexToInt(HexNum: string): LongInt;
20234: function IntToBin(Int: Integer): String;
20235: function BinToInt(Binary: String): Integer;
20236: function HexToBin(HexNum: string): string; external2
20237: function BinToHex(Binary: String): string;
20238: function IntToFloat(i: Integer): double;
20239: function AddThousandSeparator(S: string; myChr: Char): string;
20240: function Max3(const X,Y,Z: Integer): Integer;
20241: procedure Swap(var X,Y: char); // faster without inline
20242: procedure ReverseString(var S: String);
```

```
20243: function CharToHexStr(Value: Char): string;
20244: function CharToUniCode(Value: Char): string;
20245: function Hex2Dec(Value: Str002): Byte;
20246: function HexStrCodeToStr(Value: string): string;
20247: function HexToStr(i: integer; value: string): string;
20248: function UniCodeToStr(Value: string): string;
20249: function CRC16(statement: string): string;
20250: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
20251: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
20252: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
20253: Procedure ExecuteCommand(executeFile, paramstring: string);
20254: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
20255: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
20256: procedure SearchAndOpenDoc(vfilenamepath: string);
20257: procedure ShowInterfaces(myFile: string);
20258: function Fact2(av: integer): extended;
20259: Function BoolToStr(B: Boolean): string;
20260: Function GCD(x, y : LongInt) : LongInt;
20261: function LCM(m,n: longint): longint;
20262: function GetASCII: string;
20263: function GetItemHeight(Font: TFont): Integer;
20264: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
20265: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
20266: function getHINSTANCE: longword;
20267: function getHMODULE: longword;
20268: function GetASCII: string;
20269: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
20270: function WordIsOk(const AWord: string; var VW: Word): boolean;
20271: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
20272: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
20273: function SafeStr(const s: string): string;
20274: function ExtractUrlPath(const FileName: string): string;
20275: function ExtractUrlName(const FileName: string): string;
20276: function IsInternet: boolean;
20277: function RotateLeft1Bit_u32( Value: uint32): uint32;
20278: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats: Bool);
20279: procedure getEnvironmentInfo;
20280: procedure AntiFreeze;
20281: function GetCPUSpeed: Double;
20282: function IsVirtualPcGuest : Boolean;
20283: function IsVmWareGuest : Boolean;
20284: procedure StartSerialDialog;
20285: function IsWoW64: boolean;
20286: function IsWow64String(var s: string): Boolean;
20287: procedure StartThreadDemo;
20288: Function RGB(R,G,B: Byte): TColor;
20289: Function Sendln(amess: string): boolean;
20290: Procedure maXbox;
20291: Function AspectRatio(aWidth, aHeight: Integer): String;
20292: function wget(aURL, afile: string): boolean;
20293: procedure PrintList(Value: TStringList);
20294: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
20295: procedure getEnvironmentInfo;
20296: procedure AntiFreeze;
20297: function getBitmap(apath: string): TBitmap;
20298: procedure ShowMessageBig(const aText : string);
20299: function YesNoDialog(const ACaption, AMsg: string): boolean;
20300: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
20301: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
20302: //function myStrToBytes(const Value: String): TBytes;
20303: //function myBytesToStr(const Value: TBytes): String;
20304: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20305: function getBitmap(apath: string): TBitmap;
20306: procedure ShowMessageBig(const aText : string);
20307: Function StrToBytes(const Value: String): TBytes;
20308: Function BytesToStr(const Value: TBytes): String;
20309: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20310: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
20311: function FindInPaths(const fileName, paths : String) : String;
20312: procedure initHexArray(var hexn: THexArray);
20313: function josephusG(n,k: integer; var graphout: string): integer;
20314: function isPowerof2(num: int64): boolean;
20315: function powerOf2(exponent: integer): int64;
20316: function getBigPI: string;
20317: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
20318:  function GetASCIILine: string;
20319: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
20320:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
20321: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
20322: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
20323: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
20324: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
20325: function isKeypressed: boolean;
20326: function Keypress: boolean;
20327: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
20328: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
20329: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
```

```
20330: function GetOSName: string;
20331: function GetOSVersion: string;
20332: function GetOSNumber: string;
20333: function getEnvironmentString: string;
20334: procedure StrReplace(var Str: String; Old, New: String);
20335: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
20336: function getTeamViewerID: string;
20337: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
20338: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
20339: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
20340: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
20341: function StartSocketService: Boolean;
20342: procedure StartSocketServiceForm;
20343: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
20344: function GetFileList1(apath: string): TStringlist;
20345: procedure LetFileList(FileList: TStringlist; apath: string);
20346: procedure StartWeb(aurl: string);
20347: function GetTodayFiles(startdir, amask: string): TStringlist;
20348: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
20349: function JavahashCode(val: string): Integer;
20350: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
20351: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
20352: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
20353: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
20354: Procedure ConvertToGray(Cnv: TCanvas);
20355: function GetFileDate(aFile:string; aWithTime:Boolean):string;
20356: procedure ShowMemory;
20357: function ShowMemory2: string;
20358: function getHostIP: string;
20359: procedure ShowBitmap(bmap: TBitmap);
20360: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
20361: function CreateDBGridForm(dblist: TStringList): TListbox;
20362: function isService: boolean;
20363: function isApplication: boolean;
20364: function isTerminalSession: boolean;
20365:  function SetPrivilege(privilegeName: string; enable: boolean): boolean;
20366:   procedure getScriptandRunAsk;
20367:   procedure getScriptandRun(ascript: string);
20368:   function VersionCheckAct: string;
20369:   procedure getBox(aurl, extension: string);
20370:   function CheckBox: string;
20371:   function isNTFS: boolean;
20372:   //procedure doWebCamPic;
20373:   procedure doWebCamPic(picname: string);
20374:   function readm: string;
20375:   procedure getGEOMapandRunAsk;
20376:  function GetMapX(C_form,apath: string; const Data: string): boolean;
20377:   procedure GetGEOMap(C_form,apath: string; const Data: string);
20378:   function GetMapXGeoReverse(C_form: string; const lat,long: string): string;
20379:   //function RoundTo(const AValue: Extended;
20380:     //             const ADigit: TRoundToEXRangeExtended): Extended;
20381:  function DownloadFile(SourceFile, DestFile: string): Boolean;
20382:  function DownloadFileOpen(SourceFile, DestFile: string): Boolean;
20383:  function OpenMap(const Data: string): boolean;
20384:  function GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
20385:  Function getFileCount(amask: string): integer;
20386:  function CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TNavPos): string;
20387:  procedure Debugln(DebugLOGFILE: string; E: string);
20388:  function IntToFloat(i: Integer): double;
20389:  function AddThousandSeparator(S: string; myChr: Char): string;
20390:  function mymciSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
20391:
20392:
20393:
20394: // News of 3.9.8 up
20395: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20396: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20397: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20398: JvChart - TJvChart Component - 2009 Public
20399: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
20400: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
20401: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
20402: DMath DLL included incl. Demos
20403: Interface Navigator menu/View/Intf Navigator
20404: Unit Explorer menu/Debug/Units Explorer
20405: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
20406: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
20407: Script History to 9 Files WebServer light /Options/Addons/WebServer
20408: Full Text Finder, JVSimLogic Simulator Package
20409: Halt-Stop Program in Menu, WebServer2, Stop Event ,
20410: Conversion Routines, Prebuild Forms, CodeSearch
20411: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20412: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20413: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20414: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
20415: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
20416: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
20417: IDE Reflection API, Session Service Shell S3
20418: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
```

```
20419: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
20420: arduino map() function, PMRandom Generator
20421: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
20422: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
20423: REST Test Lib, Multilang Component, Forth Interpreter
20424: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
20425: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
20426: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20427: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20428: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
20429: QRCode Service, add more CFunctions like CDateTime of Synapse
20430: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
20431: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
20432: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
20433: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
20434: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
20435: BOLD Package, Indy Package5, maTRIx. MATHEMAX
20436: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
20437: emax layers: system-package-component-unit-class-function-block
20438: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
20439: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
20440: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
20441: OpenGL Game Demo: ..Options/Add Ons/Reversi
20442: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
20443: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
20444: 7% performance gain (hot spot profiling)
20445: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
20446: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
20447: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
20448: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
20449:
20450: add routines in 3.9.7.5
20451: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
20452: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
20453: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
20454: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
20455: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
20456: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
20457: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
20458:
20459: ////////////////////////// TestUnits //////////////////////////
20460:   SelftestPEM;
20461:   SelfTestCFundamentUtils;
20462:   SelfTestCFileUtils;
20463:   SelfTestCDateTime;
20464:   SelfTestCTimer;
20465:   SelfTestCRandom;
20466:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
20467:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
20468:
20469:   // Note: There's no need for installing a client certificate in the
20470:   //       webbrowser. The server asks the webbrowser to send a certificate but
20471:   //       if nothing is installed the software will work because the server
20472:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
20473:   //       file: c_cacert.p12  password: c_cakey
20474:
20475:   TGraphicControl = class(TControl)
20476:   private
20477:     FCanvas: TCanvas;
20478:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20479:   protected
20480:     procedure Paint; virtual;
20481:     property Canvas: TCanvas read FCanvas;
20482:   public
20483:     constructor Create(AOwner: TComponent); override;
20484:     destructor Destroy; override;
20485:   end;
20486:
20487:   TCustomControl = class(TWinControl)
20488:   private
20489:     FCanvas: TCanvas;
20490:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20491:   protected
20492:     procedure Paint; virtual;
20493:     procedure PaintWindow(DC: HDC); override;
20494:     property Canvas: TCanvas read FCanvas;
20495:   public
20496:     constructor Create(AOwner: TComponent); override;
20497:     destructor Destroy; override;
20498:   end;
20499:     RegisterPublishedProperties;
20500:     ('ONCHANGE', 'TNotifyEvent', iptrw);
20501:     ('ONCLICK', 'TNotifyEvent', iptrw);
20502:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
20503:     ('ONENTER', 'TNotifyEvent', iptrw);
20504:     ('ONEXIT', 'TNotifyEvent', iptrw);
20505:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
20506:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
20507:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
```

```
20508:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
20509:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
20510: //*********************************************************************
20511:   // To stop the while loop, click on Options/Show Include (boolean switch)!
20512:   Control a loop in a script with a form event:
20513:   IncludeON;   //control the while loop
20514:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
20515:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
20516:
20517: //---------------------------------------------------------------------------
20518: //**************mX4 ini-file Configuration*************************************
20519: //---------------------------------------------------------------------------
20520:   using config file maxboxdef.ini       menu/Help/Config File
20521:
20522: //*** Definitions for maXbox mX3 ***
20523: [FORM]
20524: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
20525: FONTSIZE=14
20526: EXTENSION=txt
20527: SCREENX=1386
20528: SCREENY=1077
20529: MEMHEIGHT=350
20530: PRINTFONT=Courier New //GUI Settings
20531: LINENUMBERS=Y   //line numbers at gutter in editor at left side
20532: EXCEPTIONLOG=Y  //store excepts + success in 2 log files see below! -menu Debug/Show Last Exceptions
20533: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
20534: BOOTSCRIPT=Y    //enabling load a boot script
20535: MEMORYREPORT=Y  //shows memory report on closing maXbox
20536: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
20537: NAVIGATOR=N     //shows function list at the right side of editor
20538: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
20539: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
20540: [WEB]
20541: IPPORT=8080     //for internal webserver - menu /Options/Add Ons/WebServer
20542: IPHOST=192.168.1.53
20543: ROOTCERT=filepathY
20544: SCERT=filepathY
20545: RSAKEY=filepathY
20546: VERSIONCHECK=Y
20547: APP=C:\WINDOWS\System32\calc.exe  //set path to an external app
20548:
20549:
20550: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
20551: Also possible to set report memory in script to override ini setting
20552: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
20553:
20554: After Change the ini file you can reload the file with ../Help/Config Update
20555:
20556: //---------------------------------------------------------------------------
20557: //**************mX4 maildef.ini ini-file Configuration*********************
20558: //---------------------------------------------------------------------------
20559: //*** Definitions for maXMail ***
20560: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
20561: [MAXMAIL]
20562: HOST=getmail.softwareschule.ch
20563: USER=mailusername
20564: PASS=password
20565: PORT=110
20566: SSL=Y
20567: BODY=Y
20568: LAST=5
20569:
20570: ADO Connection String:
20571: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
20572: \452_dbtreeview2access.txt
20573: program TestDbTreeViewMainForm2_ACCESS;
20574:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20575:                    +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20576:
20577: OpenSSL Lib:  unit ssl_openssl_lib;
20578: {$IFDEF CIL}
20579: const
20580:   {$IFDEF LINUX}
20581:   DLLSSLName = 'libssl.so';
20582:   DLLUtilName = 'libcrypto.so';
20583:   {$ELSE}
20584:   DLLSSLName = 'ssleay32.dll';
20585:   DLLUtilName = 'libeay32.dll';
20586:   {$ENDIF}
20587: {$ELSE}
20588: var
20589:   {$IFNDEF MSWINDOWS}
20590:     {$IFDEF DARWIN}
20591:     DLLSSLName: string = 'libssl.dylib';
20592:     DLLUtilName: string = 'libcrypto.dylib';
20593:     {$ELSE}
20594:     DLLSSLName: string = 'libssl.so';
20595:     DLLUtilName: string = 'libcrypto.so';
20596:     {$ENDIF}
```

```
20597:   {$ELSE}
20598:   DLLSSLName: string = 'ssleay32.dll';
20599:   DLLSSLName2: string = 'libssl32.dll';
20600:   DLLUtilName: string = 'libeay32.dll';
20601:   {$ENDIF}
20602: {$ENDIF}
20603:
20604:
20605: //-------------------------------------------------------------------------
20606: //*************mX4 Macro Tags  ********************************************
20607: //-------------------------------------------------------- ----------------
20608:
20609:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
       E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
20610:
20611: //Tag Macros
20612:
20613:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20614:
20615: //Tag Macros
20616: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20617: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20618: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
20619: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20620: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20621: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
20622: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20623: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
20624: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
20625:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
20626: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
20627: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20628:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20629: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
20630:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
20631: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
20632:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
20633:
20634: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20635:
20636: //Replace Macros
20637:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20638:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20639:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20640:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
20641:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
20642:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
20643:
20644:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20645:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
20646: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20647:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
20648:
20649: //-------------------------------------------------------------------------
20650: //*************mX4 ToDo List Tags  ../Help/ToDo List************************
20651: //-------------------------------------------------------- ----------------
20652:
20653:     while I < sl.Count do begin
20654: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
20655:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
20656:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
20657:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
20658:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20659:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
20660:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20661:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
20662:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20663:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
20664:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
20665:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
20666:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
20667:       Inc(I);
20668:     end;
20669:
20670: //-------------------------------------------------------------------------
20671: //*************mX4 Public  Tools API ***************************************
20672: //-------------------------------------------------------------------------
20673:   file : unit uPSI_fMain.pas;              {$OTAP} Open Tools API Catalog
20674:  // Those functions concern the editor and preprocessor, all of the IDE
20675:  Example: Call it with maxform1.Info1Click(self)
20676:  Note: Call all Methods with maxForm1., e.g.:
20677:                        maxForm1.ShellStyle1Click(self);
20678:
20679: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20680: begin
20681:  Const('BYTECODE','String 'bytecode.txt'
20682:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
20683:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
20684:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
```

```
20685:  Const('PSINC','String PS Includes (*.inc)|*.INC
20686:  Const('DEFFILENAME','String 'firstdemo.txt
20687:  Const('DEFINIFILE','String 'maxboxdef.ini
20688:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
20689:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
20690:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
20691:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
20692:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
20693:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
20694:  Const('INCLUDEBOX','String 'pas_includebox.inc
20695:  Const('BOOTSCRIPT','String 'maxbootscript.txt
20696:  Const('MBVERSION','String '3.9.9.120
20697:  Const('VERSION','String'3.9.9.120
20698:  Const('MBVER','String '399
20699:  Const('MBVERI','Integer'(399);
20700:  Const('MBVERIALL','Integer'(399120);
20701:  Const('EXENAME','String 'maXbox3.exe
20702:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
20703:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
20704:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
20705:  Const('MXINTERNETCHECK','String 'www.ask.com
20706:  Const('MXMAIL','String 'max@kleiner.com
20707:  Const('TAB','Char #$09);
20708:  Const('CODECOMPLETION','String 'bds_delphi.dci
20709:  SIRegister_TMaxForm1(CL);
20710: end;
20711:
20712:   with FindClass('TForm'),'TMaxForm1') do begin
20713:     memo2', 'TMemo', iptrw);
20714:     memo1', 'TSynMemo', iptrw);
20715:     CB1SCList', 'TComboBox', iptrw);
20716:     mxNavigator', 'TComboBox', iptrw);
20717:     IPHost', 'string', iptrw);
20718:     IPPort', 'integer', iptrw);
20719:     COMPort', 'integer', iptrw);      //3.9.6.4
20720:     Splitter1', 'TSplitter', iptrw);
20721:     PSScript', 'TPSScript', iptrw);
20722:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
20723:     MainMenu1', 'TMainMenu', iptrw);
20724:     Program1', 'TMenuItem', iptrw);
20725:     Compile1', 'TMenuItem', iptrw);
20726:     Files1', 'TMenuItem', iptrw);
20727:     open1', 'TMenuItem', iptrw);
20728:     Save2', 'TMenuItem', iptrw);
20729:     Options1', 'TMenuItem', iptrw);
20730:     Savebefore1', 'TMenuItem', iptrw);
20731:     Largefont1', 'TMenuItem', iptrw);
20732:     sBytecode1', 'TMenuItem', iptrw);
20733:     Saveas3', 'TMenuItem', iptrw);
20734:     Clear1', 'TMenuItem', iptrw);
20735:     Slinenumbers1', 'TMenuItem', iptrw);
20736:     About1', 'TMenuItem', iptrw);
20737:     Search1', 'TMenuItem', iptrw);
20738:     SynPasSyn1', 'TSynPasSyn', iptrw);
20739:     memo1', 'TSynMemo', iptrw);
20740:     SynEditSearch1', 'TSynEditSearch', iptrw);
20741:     WordWrap1', 'TMenuItem', iptrw);
20742:     XPManifest1', 'TXPManifest', iptrw);
20743:     SearchNext1', 'TMenuItem', iptrw);
20744:     Replace1', 'TMenuItem', iptrw);
20745:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
20746:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
20747:     ShowInclude1', 'TMenuItem', iptrw);
20748:     SynEditPrint1', 'TSynEditPrint', iptrw);
20749:     Printout1', 'TMenuItem', iptrw);
20750:     mnPrintColors1', 'TMenuItem', iptrw);
20751:     dlgFilePrint', 'TPrintDialog', iptrw);
20752:     dlgPrintFont1', 'TFontDialog', iptrw);
20753:     mnuPrintFont1', 'TMenuItem', iptrw);
20754:     Include1', 'TMenuItem', iptrw);
20755:     CodeCompletionList1', 'TMenuItem', iptrw);
20756:     IncludeList1', 'TMenuItem', iptrw);
20757:     ImageList1', 'TImageList', iptrw);
20758:     ImageList2', 'TImageList', iptrw);
20759:     CoolBar1', 'TCoolBar', iptrw);
20760:     ToolBar1', 'TToolBar', iptrw);
20761:     tbtnLoad', 'TToolButton', iptrw);
20762:     ToolButton2', 'TToolButton', iptrw);
20763:     tbtnFind', 'TToolButton', iptrw);
20764:     tbtnCompile', 'TToolButton', iptrw);
20765:     tbtnTrans', 'TToolButton', iptrw);
20766:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
20767:     toolbtnTutorial', 'TToolButton', iptrw);
20768:     tbtn6res', 'TToolButton', iptrw);
20769:     ToolButton5', 'TToolButton', iptrw);
20770:     ToolButton1', 'TToolButton', iptrw);
20771:     ToolButton3', 'TToolButton', iptrw);
20772:     statusBar1', 'TStatusBar', iptrw);
20773:     SaveOutput1', 'TMenuItem', iptrw);
```

```
20774:     ExportClipboard1', 'TMenuItem', iptrw);
20775:     Close1', 'TMenuItem', iptrw);
20776:     Manual1', 'TMenuItem', iptrw);
20777:     About2', 'TMenuItem', iptrw);
20778:     loadLastfile1', 'TMenuItem', iptrw);
20779:     imglogo', 'TImage', iptrw);
20780:     cedebug', 'TPSScriptDebugger', iptrw);
20781:     debugPopupMenu1', 'TPopupMenu', iptrw);
20782:     BreakPointMenu', 'TMenuItem', iptrw);
20783:     Decompile1', 'TMenuItem', iptrw);
20784:     StepInto1', 'TMenuItem', iptrw);
20785:     StepOut1', 'TMenuItem', iptrw);
20786:     Reset1', 'TMenuItem', iptrw);
20787:     DebugRun1', 'TMenuItem', iptrw);
20788:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20789:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20790:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
20791:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20792:     tutorial4', 'TMenuItem', iptrw);
20793:     ExporttoClipboard1', 'TMenuItem', iptrw);
20794:     ImportfromClipboard1', 'TMenuItem', iptrw);
20795:     N4', 'TMenuItem', iptrw);
20796:     N5', 'TMenuItem', iptrw);
20797:     N6', 'TMenuItem', iptrw);
20798:     ImportfromClipboard2', 'TMenuItem', iptrw);
20799:     tutorial1', 'TMenuItem', iptrw);
20800:     N7', 'TMenuItem', iptrw);
20801:     ShowSpecChars1', 'TMenuItem', iptrw);
20802:     OpenDirectory1', 'TMenuItem', iptrw);
20803:     procMess', 'TMenuItem', iptrw);
20804:     tbtnUseCase', 'TToolButton', iptrw);
20805:     ToolButton7', 'TToolButton', iptrw);
20806:     EditFont1', 'TMenuItem', iptrw);
20807:     UseCase1', 'TMenuItem', iptrw);
20808:     tutorial21', 'TMenuItem', iptrw);
20809:     OpenUseCase1', 'TMenuItem', iptrw);
20810:     PSImport_DB1', 'TPSImport_DB', iptrw);
20811:     tutorial31', 'TMenuItem', iptrw);
20812:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20813:     HTMLSyntax1', 'TMenuItem', iptrw);
20814:     ShowInterfaces1', 'TMenuItem', iptrw);
20815:     Tutorial5', 'TMenuItem', iptrw);
20816:     AllFunctionsList1', 'TMenuItem', iptrw);
20817:     ShowLastException1', 'TMenuItem', iptrw);
20818:     PlayMP31', 'TMenuItem', iptrw);
20819:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
20820:     texSyntax1', 'TMenuItem', iptrw);
20821:     N8', 'TMenuItem', iptrw);
20822:     GetEMails1', 'TMenuItem', iptrw);
20823:     SynCppSyn1', 'TSynCppSyn', iptrw);
20824:     CSyntax1', 'TMenuItem', iptrw);
20825:     Tutorial6', 'TMenuItem', iptrw);
20826:     New1', 'TMenuItem', iptrw);
20827:     AllObjectsList1', 'TMenuItem', iptrw);
20828:     LoadBytecode1', 'TMenuItem', iptrw);
20829:     CipherFile1', 'TMenuItem', iptrw);
20830:     N9', 'TMenuItem', iptrw);
20831:     N10', 'TMenuItem', iptrw);
20832:     Tutorial11', 'TMenuItem', iptrw);
20833:     Tutorial71', 'TMenuItem', iptrw);
20834:     UpdateService1', 'TMenuItem', iptrw);
20835:     PascalSchool1', 'TMenuItem', iptrw);
20836:     Tutorial81', 'TMenuItem', iptrw);
20837:     DelphiSite1', 'TMenuItem', iptrw);
20838:     Output1', 'TMenuItem', iptrw);
20839:     TerminalStyle1', 'TMenuItem', iptrw);
20840:     ReadOnly1', 'TMenuItem', iptrw);
20841:     ShellStyle1', 'TMenuItem', iptrw);
20842:     BigScreen1', 'TMenuItem', iptrw);
20843:     Tutorial91', 'TMenuItem', iptrw);
20844:     SaveOutput2', 'TMenuItem', iptrw);
20845:     N11', 'TMenuItem', iptrw);
20846:     SaveScreenshot', 'TMenuItem', iptrw);
20847:     Tutorial101', 'TMenuItem', iptrw);
20848:     SQLSyntax1', 'TMenuItem', iptrw);
20849:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
20850:     Console1', 'TMenuItem', iptrw);
20851:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
20852:     XMLSyntax1', 'TMenuItem', iptrw);
20853:     ComponentCount1', 'TMenuItem', iptrw);
20854:     NewInstance1', 'TMenuItem', iptrw);
20855:     toolbtnTutorial', 'TToolButton', iptrw);
20856:     Memory1', 'TMenuItem', iptrw);
20857:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
20858:     JavaSyntax1', 'TMenuItem', iptrw);
20859:     SyntaxCheck1', 'TMenuItem', iptrw);
20860:     Tutorial10Statistics1', 'TMenuItem', iptrw);
20861:     ScriptExplorer1', 'TMenuItem', iptrw);
20862:     FormOutput1', 'TMenuItem', iptrw);
```

```
20863:     ArduinoDump1', 'TMenuItem', iptrw);
20864:     AndroidDump1', 'TMenuItem', iptrw);
20865:     GotoEnd1', 'TMenuItem', iptrw);
20866:     AllResourceList1', 'TMenuItem', iptrw);
20867:     ToolButton4', 'TToolButton', iptrw);
20868:     tbtn6res', 'TToolButton', iptrw);
20869:     Tutorial11Forms1', 'TMenuItem', iptrw);
20870:     Tutorial12SQL1', 'TMenuItem', iptrw);
20871:     ResourceExplore1', 'TMenuItem', iptrw);
20872:     Info1', 'TMenuItem', iptrw);
20873:     N12', 'TMenuItem', iptrw);
20874:     CryptoBox1', 'TMenuItem', iptrw);
20875:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
20876:     CipherFile2', 'TMenuItem', iptrw);
20877:     N13', 'TMenuItem', iptrw);
20878:     ModulesCount1', 'TMenuItem', iptrw);
20879:     AddOns2', 'TMenuItem', iptrw);
20880:     N4GewinntGame1', 'TMenuItem', iptrw);
20881:     DocuforAddOns1', 'TMenuItem', iptrw);
20882:     Tutorial14Async1', 'TMenuItem', iptrw);
20883:     Lessons15Review1', 'TMenuItem', iptrw);
20884:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
20885:     PHPSyntax1', 'TMenuItem', iptrw);
20886:     Breakpoint1', 'TMenuItem', iptrw);
20887:     SerialRS2321', 'TMenuItem', iptrw);
20888:     N14', 'TMenuItem', iptrw);
20889:     SynCSSyn1', 'TSynCSSyn', iptrw);
20890:     CSyntax2', 'TMenuItem', iptrw);
20891:     Calculator1', 'TMenuItem', iptrw);
20892:     tbtnSerial', 'TToolButton', iptrw);
20893:     ToolButton8', 'TToolButton', iptrw);
20894:     Tutorial151', 'TMenuItem', iptrw);
20895:     N15', 'TMenuItem', iptrw);
20896:     N16', 'TMenuItem', iptrw);
20897:     ControlBar1', 'TControlBar', iptrw);
20898:     ToolBar2', 'TToolBar', iptrw);
20899:     BtnOpen', 'TToolButton', iptrw);
20900:     BtnSave', 'TToolButton', iptrw);
20901:     BtnPrint', 'TToolButton', iptrw);
20902:     BtnColors', 'TToolButton', iptrw);
20903:     btnClassReport', 'TToolButton', iptrw);
20904:     BtnRotateRight', 'TToolButton', iptrw);
20905:     BtnFullSize', 'TToolButton', iptrw);
20906:     BtnFitToWindowSize', 'TToolButton', iptrw);
20907:     BtnZoomMinus', 'TToolButton', iptrw);
20908:     BtnZoomPlus', 'TToolButton', iptrw);
20909:     Panel1', 'TPanel', iptrw);
20910:     LabelBrettgroesse', 'TLabel', iptrw);
20911:     CB1SCList', 'TComboBox', iptrw);
20912:     ImageListNormal', 'TImageList', iptrw);
20913:     spbtnexplore', 'TSpeedButton', iptrw);
20914:     spbtnexample', 'TSpeedButton', iptrw);
20915:     spbsaveas', 'TSpeedButton', iptrw);
20916:     imglogobox', 'TImage', iptrw);
20917:     EnlargeFont1', 'TMenuItem', iptrw);
20918:     EnlargeFont2', 'TMenuItem', iptrw);
20919:     ShrinkFont1', 'TMenuItem', iptrw);
20920:     ThreadDemo1', 'TMenuItem', iptrw);
20921:     HEXEditor1', 'TMenuItem', iptrw);
20922:     HEXView1', 'TMenuItem', iptrw);
20923:     HEXInspect1', 'TMenuItem', iptrw);
20924:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
20925:     ExporttoHTML1', 'TMenuItem', iptrw);
20926:     ClassCount1', 'TMenuItem', iptrw);
20927:     HTMLOutput1', 'TMenuItem', iptrw);
20928:     HEXEditor2', 'TMenuItem', iptrw);
20929:     Minesweeper1', 'TMenuItem', iptrw);
20930:     N17', 'TMenuItem', iptrw);
20931:     PicturePuzzle1', 'TMenuItem', iptrw);
20932:     sbvclhelp', 'TSpeedButton', iptrw);
20933:     DependencyWalker1', 'TMenuItem', iptrw);
20934:     WebScanner1', 'TMenuItem', iptrw);
20935:     View1', 'TMenuItem', iptrw);
20936:     mnToolbar1', 'TMenuItem', iptrw);
20937:     mnStatusbar2', 'TMenuItem', iptrw);
20938:     mnConsole2', 'TMenuItem', iptrw);
20939:     mnCoolbar2', 'TMenuItem', iptrw);
20940:     mnSplitter2', 'TMenuItem', iptrw);
20941:     WebServer1', 'TMenuItem', iptrw);
20942:     Tutorial17Server1', 'TMenuItem', iptrw);
20943:     Tutorial18Arduino1', 'TMenuItem', iptrw);
20944:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
20945:     PerlSyntax1', 'TMenuItem', iptrw);
20946:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
20947:     PythonSyntax1', 'TMenuItem', iptrw);
20948:     DMathLibrary1', 'TMenuItem', iptrw);
20949:     IntfNavigator1', 'TMenuItem', iptrw);
20950:     EnlargeFontConsole1', 'TMenuItem', iptrw);
20951:     ShrinkFontConsole1', 'TMenuItem', iptrw);
```

```
20952:     SetInterfaceList1', 'TMenuItem', iptrw);
20953:     popintfList', 'TPopupMenu', iptrw);
20954:     intfAdd1', 'TMenuItem', iptrw);
20955:     intfDelete1', 'TMenuItem', iptrw);
20956:     intfRefactor1', 'TMenuItem', iptrw);
20957:     Defactor1', 'TMenuItem', iptrw);
20958:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
20959:     Tutorial20Regex', 'TMenuItem', iptrw);
20960:     N18', 'TMenuItem', iptrw);
20961:     ManualE1', 'TMenuItem', iptrw);
20962:     FullTextFinder1', 'TMenuItem', iptrw);
20963:     Move1', 'TMenuItem', iptrw);
20964:     FractalDemo1', 'TMenuItem', iptrw);
20965:     Tutorial21Android1', 'TMenuItem', iptrw);
20966:     Tutorial0Function1', 'TMenuItem', iptrw);
20967:     SimuLogBox1', 'TMenuItem', iptrw);
20968:     OpenExamples1', 'TMenuItem', iptrw);
20969:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
20970:     JavaScriptSyntax1', 'TMenuItem', iptrw);
20971:     Halt1', 'TMenuItem', iptrw);
20972:     CodeSearch1', 'TMenuItem', iptrw);
20973:     SynRubySyn1', 'TSynRubySyn', iptrw);
20974:     RubySyntax1', 'TMenuItem', iptrw);
20975:     Undo1', 'TMenuItem', iptrw);
20976:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
20977:     LinuxShellScript1', 'TMenuItem', iptrw);
20978:     Rename1', 'TMenuItem', iptrw);
20979:     spdcodesearch', 'TSpeedButton', iptrw);
20980:     Preview1', 'TMenuItem', iptrw);
20981:     Tutorial22Services1', 'TMenuItem', iptrw);
20982:     Tutorial23RealTime1', 'TMenuItem', iptrw);
20983:     Configuration1', 'TMenuItem', iptrw);
20984:     MP3Player1', 'TMenuItem', iptrw);
20985:     DLLSpy1', 'TMenuItem', iptrw);
20986:     SynURIOpener1', 'TSynURIOpener', iptrw);
20987:     SynURISyn1', 'TSynURISyn', iptrw);
20988:     URILinksClicks1', 'TMenuItem', iptrw);
20989:     EditReplace1', 'TMenuItem', iptrw);
20990:     GotoLine1', 'TMenuItem', iptrw);
20991:     ActiveLineColor1', 'TMenuItem', iptrw);
20992:     ConfigFile1', 'TMenuItem', iptrw);
20993:     Sort1Intflist', 'TMenuItem', iptrw);
20994:     Redo1', 'TMenuItem', iptrw);
20995:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
20996:     Tutorial25Configuration1', 'TMenuItem', iptrw);
20997:     IndentSelection1', 'TMenuItem', iptrw);
20998:     UnindentSection1', 'TMenuItem', iptrw);
20999:     SkyStyle1', 'TMenuItem', iptrw);
21000:     N19', 'TMenuItem', iptrw);
21001:     CountWords1', 'TMenuItem', iptrw);
21002:     imbookmarkimages', 'TImageList', iptrw);
21003:     Bookmark11', 'TMenuItem', iptrw);
21004:     N20', 'TMenuItem', iptrw);
21005:     Bookmark21', 'TMenuItem', iptrw);
21006:     Bookmark31', 'TMenuItem', iptrw);
21007:     Bookmark41', 'TMenuItem', iptrw);
21008:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
21009:
21010:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
21011:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
21012:     Procedure PSScriptCompile( Sender : TPSScript)
21013:     Procedure Compile1Click( Sender : TObject)
21014:     Procedure PSScriptExecute( Sender : TPSScript)
21015:     Procedure open1Click( Sender : TObject)
21016:     Procedure Save2Click( Sender : TObject)
21017:     Procedure Savebefore1Click( Sender : TObject)
21018:     Procedure Largefont1Click( Sender : TObject)
21019:     Procedure FormActivate( Sender : TObject)
21020:     Procedure SBytecode1Click( Sender : TObject)
21021:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
21022:     Procedure Saveas3Click( Sender : TObject)
21023:     Procedure Clear1Click( Sender : TObject)
21024:     Procedure Slinenumbers1Click( Sender : TObject)
21025:     Procedure About1Click( Sender : TObject)
21026:     Procedure Search1Click( Sender : TObject)
21027:     Procedure FormCreate( Sender : TObject)
21028:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
21029:                                                    var Action : TSynReplaceAction)
21030:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
21031:     Procedure WordWrap1Click( Sender : TObject)
21032:     Procedure SearchNext1Click( Sender : TObject)
21033:     Procedure Replace1Click( Sender : TObject)
21034:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
21035:     Procedure ShowInclude1Click( Sender : TObject)
21036:     Procedure Printout1Click( Sender : TObject)
21037:     Procedure mnuPrintFont1Click( Sender : TObject)
21038:     Procedure Include1Click( Sender : TObject)
21039:     Procedure FormDestroy( Sender : TObject)
21040:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
```

```
21041:     Procedure UpdateView1Click( Sender : TObject)
21042:     Procedure CodeCompletionList1Click( Sender : TObject)
21043:     Procedure SaveOutput1Click( Sender : TObject)
21044:     Procedure ExportClipboard1Click( Sender : TObject)
21045:     Procedure Close1Click( Sender : TObject)
21046:     Procedure Manual1Click( Sender : TObject)
21047:     Procedure LoadLastFile1Click( Sender : TObject)
21048:     Procedure Memo1Change( Sender : TObject)
21049:     Procedure Decompile1Click( Sender : TObject)
21050:     Procedure StepInto1Click( Sender : TObject)
21051:     Procedure StepOut1Click( Sender : TObject)
21052:     Procedure Reset1Click( Sender : TObject)
21053:     Procedure cedebugAfterExecute( Sender : TPSScript)
21054:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
21055:     Procedure cedebugCompile( Sender : TPSScript)
21056:     Procedure cedebugExecute( Sender : TPSScript)
21057:     Procedure cedebugIdle( Sender : TObject)
21058:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
21059:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
21060:     Procedure BreakPointMenuClick( Sender : TObject)
21061:     Procedure DebugRun1Click( Sender : TObject)
21062:     Procedure tutorial4Click( Sender : TObject)
21063:     Procedure ImportfromClipboard1Click( Sender : TObject)
21064:     Procedure ImportfromClipboard2Click( Sender : TObject)
21065:     Procedure tutorial1Click( Sender : TObject)
21066:     Procedure ShowSpecChars1Click( Sender : TObject)
21067:     Procedure StatusBar1DblClick( Sender : TObject)
21068:     Procedure PSScriptLine( Sender : TObject)
21069:     Procedure OpenDirectory1Click( Sender : TObject)
21070:     Procedure procMessClick( Sender : TObject)
21071:     Procedure tbtnUseCaseClick( Sender : TObject)
21072:     Procedure EditFont1Click( Sender : TObject)
21073:     Procedure tutorial21Click( Sender : TObject)
21074:     Procedure tutorial31Click( Sender : TObject)
21075:     Procedure HTMLSyntax1Click( Sender : TObject)
21076:     Procedure ShowInterfaces1Click( Sender : TObject)
21077:     Procedure Tutorial5Click( Sender : TObject)
21078:     Procedure ShowLastException1Click( Sender : TObject)
21079:     Procedure PlayMP31Click( Sender : TObject)
21080:     Procedure AllFunctionsList1Click( Sender : TObject)
21081:     Procedure texSyntax1Click( Sender : TObject)
21082:     Procedure GetEMails1Click( Sender : TObject)
21083:     procedure DelphiSite1Click(Sender: TObject);
21084:     procedure TerminalStyle1Click(Sender: TObject);
21085:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
21086:     procedure ShellStyle1Click(Sender: TObject);
21087:     procedure Console1Click(Sender: TObject);     //3.2
21088:     procedure BigScreen1Click(Sender: TObject);
21089:     procedure Tutorial91Click(Sender: TObject);
21090:     procedure SaveScreenshotClick(Sender: TObject);
21091:     procedure Tutorial101Click(Sender: TObject);
21092:     procedure SQLSyntax1Click(Sender: TObject);
21093:     procedure XMLSyntax1Click(Sender: TObject);
21094:     procedure ComponentCount1Click(Sender: TObject);
21095:     procedure NewInstance1Click(Sender: TObject);
21096:     procedure CSyntax1Click(Sender: TObject);
21097:     procedure Tutorial6Click(Sender: TObject);
21098:     procedure New1Click(Sender: TObject);
21099:     procedure AllObjectsList1Click(Sender: TObject);
21100:     procedure LoadBytecode1Click(Sender: TObject);
21101:     procedure CipherFile1Click(Sender: TObject);  //V3.5
21102:     procedure NewInstance1Click(Sender: TObject);
21103:     procedure toolbtnTutorialClick(Sender: TObject);
21104:     procedure Memory1Click(Sender: TObject);
21105:     procedure JavaSyntax1Click(Sender: TObject);
21106:     procedure SyntaxCheck1Click(Sender: TObject);
21107:     procedure ScriptExplorer1Click(Sender: TObject);
21108:     procedure FormOutput1Click(Sender: TObject);  //V3.6
21109:     procedure GotoEnd1Click(Sender: TObject);
21110:     procedure AllResourceList1Click(Sender: TObject);
21111:     procedure tbtn6resClick(Sender: TObject);   //V3.7
21112:     procedure Info1Click(Sender: TObject);
21113:     procedure Tutorial10Statistics1Click(Sender: TObject);
21114:     procedure Tutorial11Forms1Click(Sender: TObject);
21115:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
21116:     procedure ResourceExplore1Click(Sender: TObject);
21117:     procedure Info1Click(Sender: TObject);
21118:     procedure CryptoBox1Click(Sender: TObject);
21119:     procedure ModulesCount1Click(Sender: TObject);
21120:     procedure N4GewinntGame1Click(Sender: TObject);
21121:     procedure PHPSyntax1Click(Sender: TObject);
21122:     procedure SerialRS2321Click(Sender: TObject);
21123:     procedure CSyntax2Click(Sender: TObject);
21124:     procedure Calculator1Click(Sender: TObject);
21125:     procedure Tutorial13Ciphering1Click(Sender: TObject);
21126:     procedure Tutorial14Async1Click(Sender: TObject);
21127:     procedure PHPSyntax1Click(Sender: TObject);
21128:     procedure BtnZoomPlusClick(Sender: TObject);
21129:     procedure BtnZoomMinusClick(Sender: TObject);
```

```
21130:     procedure btnClassReportClick(Sender: TObject);
21131:     procedure ThreadDemo1Click(Sender: TObject);
21132:     procedure HEXView1Click(Sender: TObject);
21133:     procedure ExporttoHTML1Click(Sender: TObject);
21134:     procedure Minesweeper1Click(Sender: TObject);
21135:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
21136:     procedure sbvclhelpClick(Sender: TObject);
21137:     procedure DependencyWalker1Click(Sender: TObject);
21138:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
21139:     procedure WebScanner1Click(Sender: TObject);
21140:     procedure mnToolbar1Click(Sender: TObject);
21141:     procedure mnStatusbar2Click(Sender: TObject);
21142:     procedure mnConsole2Click(Sender: TObject);
21143:     procedure mnCoolbar2Click(Sender: TObject);
21144:     procedure mnSplitter2Click(Sender: TObject);
21145:     procedure WebServer1Click(Sender: TObject);
21146:     procedure PerlSyntax1Click(Sender: TObject);
21147:     procedure PythonSyntax1Click(Sender: TObject);
21148:     procedure DMathLibrary1Click(Sender: TObject);
21149:     procedure IntfNavigator1Click(Sender: TObject);
21150:     procedure FullTextFinder1Click(Sender: TObject);
21151:     function AppName: string;
21152:     function ScriptName: string;
21153:     function LastName: string;
21154:     procedure FractalDemo1Click(Sender: TObject);
21155:     procedure SimuLogBox1Click(Sender: TObject);
21156:     procedure OpenExamples1Click(Sender: TObject);
21157:     procedure Halt1Click(Sender: TObject);
21158:     procedure Stop;
21159:     procedure CodeSearch1Click(Sender: TObject);
21160:     procedure RubySyntax1Click(Sender: TObject);
21161:     procedure Undo1Click(Sender: TObject);
21162:     procedure LinuxShellScript1Click(Sender: TObject);
21163:     procedure WebScannerDirect(urls: string);
21164:     procedure WebScanner(urls: string);
21165:     procedure LoadInterfaceList2;
21166:     procedure DLLSpy1Click(Sender: TObject);
21167:     procedure Memo1DblClick(Sender: TObject);
21168:     procedure URILinksClicks1Click(Sender: TObject);
21169:     procedure GotoLine1Click(Sender: TObject);
21170:     procedure ConfigFile1Click(Sender: TObject);
21171:     Procedure Sort1IntflistClick( Sender : TObject)
21172:     Procedure Redo1Click( Sender : TObject)
21173:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
21174:     Procedure IndentSelection1Click( Sender : TObject)
21175:     Procedure UnindentSection1Click( Sender : TObject)
21176:     Procedure SkyStyle1Click( Sender : TObject)
21177:     Procedure CountWords1Click( Sender : TObject)
21178:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
21179:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
21180:     Procedure Bookmark11Click( Sender : TObject)
21181:     Procedure Bookmark21Click( Sender : TObject)
21182:     Procedure Bookmark31Click( Sender : TObject)
21183:     Procedure Bookmark41Click( Sender : TObject)
21184:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
21185:     'STATMemoryReport', 'boolean', iptrw);
21186:     'IPPort', 'integer', iptrw);
21187:     'COMPort', 'integer', iptrw);
21188:     'lbintflist', 'TListBox', iptrw);
21189:     Function GetStatChange : boolean
21190:     Procedure SetStatChange( vstat : boolean)
21191:     Function GetActFileName : string
21192:     Procedure SetActFileName( vname : string)
21193:     Function GetLastFileName : string
21194:     Procedure SetLastFileName( vname : string)
21195:     Procedure WebScannerDirect( urls : string)
21196:     Procedure LoadInterfaceList2
21197:     Function GetStatExecuteShell : boolean
21198:     Procedure DoEditorExecuteCommand( EditorCommand : word)
21199:     function GetActiveLineColor: TColor
21200:     procedure SetActiveLineColor(acolor: TColor)
21201:     procedure ScriptListbox1Click(Sender: TObject);
21202:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
21203:     procedure EnlargeGutter1Click(Sender: TObject);
21204:     procedure Tetris1Click(Sender: TObject);
21205:     procedure ToDoList1Click(Sender: TObject);
21206:     procedure ProcessList1Click(Sender: TObject);
21207:     procedure MetricReport1Click(Sender: TObject);
21208:     procedure ProcessList1Click(Sender: TObject);
21209:     procedure TCPSockets1Click(Sender: TObject);
21210:     procedure ConfigUpdate1Click(Sender: TObject);
21211:     procedure ADOWorkbench1Click(Sender: TObject);
21212:     procedure SocketServer1Click(Sender: TObject);
21213:     procedure FormDemo1Click(Sender: TObject);
21214:     procedure Richedit1Click(Sender: TObject);
21215:     procedure SimpleBrowser1Click(Sender: TObject);
21216:     procedure DOSShell1Click(Sender: TObject);
21217:     procedure SynExport1Click(Sender: TObject);
21218:     procedure ExporttoRTF1Click(Sender: TObject);
```

```
21219:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
21220:     procedure SOAPTester1Click(Sender: TObject);
21221:     procedure Sniffer1Click(Sender: TObject);
21222:     procedure AutoDetectSyntax1Click(Sender: TObject);
21223:     procedure FPlot1Click(Sender: TObject);
21224:     procedure PasStyle1Click(Sender: TObject);
21225:     procedure Tutorial183RGBLED1Click(Sender: TObject);
21226:     procedure Reversi1Click(Sender: TObject);
21227:     procedure ManualmaXbox1Click(Sender: TObject);
21228:     procedure BlaisePascalMagazine1Click(Sender: TObject);
21229:     procedure AddToDo1Click(Sender: TObject);
21230:     procedure CreateGUID1Click(Sender: TObject);
21231:     procedure Tutorial27XML1Click(Sender: TObject);
21232:     procedure CreateDLLStub1Click(Sender: TObject);
21233:     procedure Tutorial28DLL1Click(Sender: TObject);');
21234:     procedure ResetKeyPressed;');
21235:     procedure KeyPressedFalse;
21236:     procedure FileChanges1Click(Sender: TObject);');
21237:     procedure OpenGLTry1Click(Sender: TObject);');
21238:     procedure AllUnitList1Click(Sender: TObject);');
21239:     procedure Tutorial29UMLClick(Sender: TObject);
21240:     procedure CreateHeader1Click(Sender: TObject);
21241:     procedure Oscilloscope1Click(Sender: TObject);');
21242:     procedure Tutorial30WOT1Click(Sender: TObject);');
21243:     procedure GetWebScript1Click(Sender: TObject);');
21244:     procedure Checkers1Click(Sender: TObject);');
21245:     procedure TaskMgr1Click(Sender: TObject);');
21246:     procedure WebCam1Click(Sender: TObject);');
21247:     procedure Tutorial31Closure1Click(Sender: TObject);');
21248:     procedure GEOMapView1Click(Sender: TObject);');
21249:     procedure Run1Click(Sender: TObject);
21250:     MaxForm1.GPSSatView1Click, 'GPSSatView1Click');
21251:     MaxForm1.N3DLab1Click, 'N3DLab1Click');
21252:     procedure ExternalApp1Click(Sender: TObject);');
21253:
21254:
21255: //-----------------------------------------------------------------------
21256: //**************mX4 Editor SynEdit Tools API *********************************
21257: //-----------------------------------------------------------------------
21258: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
21259: begin
21260:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
21261:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
21262:     Constructor Create( AOwner : TComponent)
21263:     SelStart', 'Integer', iptrw);
21264:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
21265:     Procedure UpdateCaret
21266:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
21267:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
21268:     Procedure BeginUndoBlock
21269:     Procedure BeginUpdate
21270:     Function CaretInView : Boolean
21271:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
21272:     Procedure Clear
21273:     Procedure ClearAll
21274:     Procedure ClearBookMark( BookMark : Integer)
21275:     Procedure ClearSelection
21276:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
21277:     Procedure ClearUndo
21278:     Procedure CopyToClipboard
21279:     Procedure CutToClipboard
21280:     Procedure DoCopyToClipboard( const SText : string)
21281:     Procedure EndUndoBlock
21282:     Procedure EndUpdate
21283:     Procedure EnsureCursorPosVisible
21284:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
21285:     Procedure FindMatchingBracket
21286:     Function GetMatchingBracket : TBufferCoord
21287:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
21288:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
21289:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
21290:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
21291:               : TSynHighlighterAttributes) : boolean
21292:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
21293:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
21294:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
21295:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
21296:     Procedure GotoBookMark( BookMark : Integer)
21297:     Procedure GotoLineAndCenter( ALine : Integer)
21298:     Function IdentChars : TSynIdentChars
21299:     Procedure InvalidateGutter
21300:     Procedure InvalidateGutterLine( aLine : integer)
21301:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
21302:     Procedure InvalidateLine( Line : integer)
21303:     Procedure InvalidateLines( FirstLine, LastLine : integer)
21304:     Procedure InvalidateSelection
21305:     Function IsBookmark( BookMark : integer) : boolean
21306:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
21307:     Procedure LockUndo
```

```
21308:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
21309:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
21310:     Function LineToRow( aLine : integer) : integer
21311:     Function RowToLine( aRow : integer) : integer
21312:     Function NextWordPos : TBufferCoord
21313:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
21314:     Procedure PasteFromClipboard
21315:     Function WordStart : TBufferCoord
21316:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
21317:     Function WordEnd : TBufferCoord
21318:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
21319:     Function PrevWordPos : TBufferCoord
21320:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
21321:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
21322:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
21323:     Procedure Redo
21324:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
21325:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
21326:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
21327:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
21328:     Procedure SelectAll
21329:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
21330:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
21331:     Procedure SetDefaultKeystrokes
21332:     Procedure SetSelWord
21333:     Procedure SetWordBlock( Value : TBufferCoord)
21334:     Procedure Undo
21335:     Procedure UnlockUndo
21336:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
21337:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
21338:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
21339:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
21340:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
21341:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
21342:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
21343:     Procedure AddFocusControl( aControl : TWinControl)
21344:     Procedure RemoveFocusControl( aControl : TWinControl)
21345:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
21346:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
21347:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
21348:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
21349:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
21350:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
21351:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
21352:     Procedure RemoveLinesPointer
21353:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
21354:     Procedure UnHookTextBuffer
21355:     BlockBegin', 'TBufferCoord', iptrw);
21356:     BlockEnd', 'TBufferCoord', iptrw);
21357:     CanPaste', 'Boolean', iptr);
21358:     CanRedo', 'boolean', iptr);
21359:     CanUndo', 'boolean', iptr);
21360:     CaretX', 'Integer', iptrw);
21361:     CaretY', 'Integer', iptrw);
21362:     CaretXY', 'TBufferCoord', iptrw);
21363:     ActiveLineColor', 'TColor', iptrw);
21364:     DisplayX', 'Integer', iptr);
21365:     DisplayY', 'Integer', iptr);
21366:     DisplayXY', 'TDisplayCoord', iptr);
21367:     DisplayLineCount', 'integer', iptr);
21368:     CharsInWindow', 'Integer', iptr);
21369:     CharWidth', 'integer', iptr);
21370:     Font', 'TFont', iptrw);
21371:     GutterWidth', 'Integer', iptr);
21372:     Highlighter', 'TSynCustomHighlighter', iptrw);
21373:     LeftChar', 'Integer', iptrw);
21374:     LineHeight', 'integer', iptr);
21375:     LinesInWindow', 'Integer', iptr);
21376:     LineText', 'string', iptrw);
21377:     Lines', 'TStrings', iptrw);
21378:     Marks', 'TSynEditMarkList', iptr);
21379:     MaxScrollWidth', 'integer', iptrw);
21380:     Modified', 'Boolean', iptrw);
21381:     PaintLock', 'Integer', iptr);
21382:     ReadOnly', 'Boolean', iptrw);
21383:     SearchEngine', 'TSynEditSearchCustom', iptrw);
21384:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
21385:     SelTabBlock', 'Boolean', iptr);
21386:     SelTabLine', 'Boolean', iptr);
21387:     SelText', 'string', iptrw);
21388:     StateFlags', 'TSynStateFlags', iptr);
21389:     Text', 'string', iptrw);
21390:     TopLine', 'Integer', iptrw);
21391:     WordAtCursor', 'string', iptr);
21392:     WordAtMouse', 'string', iptr);
21393:     UndoList', 'TSynEditUndoList', iptr);
21394:     RedoList', 'TSynEditUndoList', iptr);
21395:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
21396:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
```

```
21397:     BorderStyle', 'TSynBorderStyle', iptrw);
21398:     ExtraLineSpacing', 'integer', iptrw);
21399:     Gutter', 'TSynGutter', iptrw);
21400:     HideSelection', 'boolean', iptrw);
21401:     InsertCaret', 'TSynEditCaretType', iptrw);
21402:     InsertMode', 'boolean', iptrw);
21403:     IsScrolling', 'Boolean', iptr);
21404:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
21405:     MaxUndo', 'Integer', iptrw);
21406:     Options', 'TSynEditorOptions', iptrw);
21407:     OverwriteCaret', 'TSynEditCaretType', iptrw);
21408:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
21409:     ScrollHintColor', 'TColor', iptrw);
21410:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
21411:     ScrollBars', 'TScrollStyle', iptrw);
21412:     SelectedColor', 'TSynSelectedColor', iptrw);
21413:     SelectionMode', 'TSynSelectionMode', iptrw);
21414:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
21415:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
21416:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
21417:     WordWrapGlyph', 'TSynGlyph', iptrw);
21418:     OnChange', 'TNotifyEvent', iptrw);
21419:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
21420:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
21421:     OnContextHelp', 'TContextHelpEvent', iptrw);
21422:     OnDropFiles', 'TDropFilesEvent', iptrw);
21423:     OnGutterClick', 'TGutterClickEvent', iptrw);
21424:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
21425:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
21426:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
21427:     OnPaint', 'TPaintEvent', iptrw);
21428:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
21429:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
21430:     OnReplaceText', 'TReplaceTextEvent', iptrw);
21431:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
21432:     OnStatusChange', 'TStatusChangeEvent', iptrw);
21433:     OnPaintTransient', 'TPaintTransient', iptrw);
21434:     OnScroll', 'TScrollEvent', iptrw);
21435:   end;
21436: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
21437: Function GetPlaceableHighlighters : TSynHighlighterList
21438: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
21439: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
21440: Procedure GetEditorCommandValues( Proc : TGetStrProc)
21441: Procedure GetEditorCommandExtended( Proc : TGetStrProc)
21442: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
21443: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
21444: Function ConvertCodeStringToExtended( AString : String) : String
21445: Function ConvertExtendedToCodeString( AString : String) : String
21446: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
21447: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
21448: Function IndexToEditorCommand( const AIndex : Integer) : Integer
21449:
21450:  TSynEditorOption = (
21451:    eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
21452:    eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
21453:                               // preceding line
21454:    eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
21455:    eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
21456:                               //direction any more
21457:    eoDragDropEditing,         //Allows to select a textblock and drag it in document to another location
21458:    eoDropFiles,               //Allows the editor accept OLE file drops
21459:    eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
21460:    eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
21461:    eoGroupUndo,               //When undoing/redoing actions,handle all cont.changes same kind in onecall
21462:                               //instead undoing/redoing each command separately
21463:    eoHalfPageScroll,          //By scrolling with page-up/page-down commands,only scroll half page attime
21464:    eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
21465:    If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
21466:    eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
21467:    eoNoCaret,                 //Makes it so the caret is never visible
21468:    eoNoSelection,             //Disables selecting text
21469:    eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
21470:    eoScrollByOneLess,         //Forces scrolling to be one less
21471:    eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
21472:    eoScrollPastEof,           //Allows the cursor to go past the end of file marker
21473:    eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
21474:    eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
21475:    eoShowSpecialChars,        //Shows the special Characters
21476:    eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
21477:    eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
21478:    eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
21479:    eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
21480:    eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
21481:    eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
21482:
21483:    ********************************Important Editor Short Cuts*****************************);
21484:    Double click to select a word and count words with highlightning.
21485:    Triple click to select a line.
```

```
21486:     CTRL+SHIFT+click to extend a selection.
21487:     Drag with the ALT key down to select columns of text !!!
21488:     Drag and drop is supported.
21489:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
21490:     Type CTRL+A to select all.
21491:     Type CTRL+N to set a new line.
21492:     Type CTRL+T to delete a line or token. //Tokenizer
21493:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
21494:     Type CTRL+Shift+T to add ToDo in line and list.
21495:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
21496:     Type CTRL[0..9] to jump or get to bookmarks.
21497:     Type Home to position cursor at beginning of current line and End to position it at end of line.
21498:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
21499:     Page Up and Page Down work as expected.
21500:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
21501:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
21502:
21503: {$ Short Key Positions Ctrl<A-Z>: }
21504: def
21505:    <A> Select All
21506:    <B> Count Words
21507:    <C> Copy
21508:    <D> Internet Start
21509:    <E> Script List
21510:    <F> Find
21511:    <G> Goto
21512:    <H> Mark Line
21513:    <I> Interface List
21514:    <J> Code Completion
21515:    <K> Console
21516:    <L> Interface List Box
21517:    <M> Font Larger -
21518:    <N> New Line
21519:    <O> Open File
21520:    <P> Font Smaller +
21521:    <Q> Quit
21522:    <R> Replace
21523:    <S> Save!
21524:    <T> Delete Line
21525:    <U> Use Case Editor
21526:    <V> Paste
21527:    <W> URI Links
21528:    <X> Reserved for coding use internal
21529:    <Y> Delete Line
21530:    <Z> Undo
21531:
21532: ref
21533:    F1 Help
21534:    F2 Syntax Check
21535:    F3 Search Next
21536:    F4 New Instance
21537:    F5 Line Mark /Breakpoint
21538:    F6 Goto End
21539:    F7 Debug Step Into
21540:    F8 Debug Step Out
21541:    F9 Compile
21542:    F10 Menu
21543:    F11 Word Count Highlight
21544:    F12 Reserved for coding use internal
21545:
21546:    AddRegisteredVariable('Application', 'TApplication');
21547:    AddRegisteredVariable('Screen', 'TScreen');
21548:    AddRegisteredVariable('Self', 'TForm');
21549:    AddRegisteredVariable('Memo1', 'TSynMemo');
21550:    AddRegisteredVariable('memo2', 'TMemo');
21551:    AddRegisteredVariable('maxForm1', 'TMaxform1');  //!!
21552:    AddRegisteredVariable('debugout', 'Tdebugoutput');  //!!
21553:    AddRegisteredVariable('hlog','THotlog');  //!!
21554:    AddRegisteredVariable( it ,integer');  //for closure!!
21555:    AddRegisteredVariable( sr ,string');  //for closure
21556:    AddRegisteredVariable( bt ,boolean');  //for closure
21557:    AddRegisteredVariable( ft ,double');  //for closure
21558:    AddRegisteredVariable( srlist ,TStringlist');  //for closures
21559:
21560:  def ReservedWords: array[0..82] of string =
21561:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
21562:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
21563:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
21564:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
21565:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
21566:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
21567:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
21568:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
21569:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
21570:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
21571:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
21572:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
21573: //-----------------------------------------------------------------------------
21574: //*************End of mX4 Public  Tools API *********************************
```

```
21575: //-------------------------------------------------------------------------------
21576:
21577: Amount of Functions: 13930
21578: Amount of Procedures: 8516
21579: Amount of Constructors: 1382
21580: Totals of Calls: 23828
21581: SHA1: Win of 3.9.9.120 3EFB12E5729BDAA745373C2743F9DD1E93C9295D
21582:
21583:
21584: ********************************************************
21585:  Doc Short Manual with 50 Tips!
21586: ********************************************************
21587: - Install: just save your maxboxdef.ini before and then extract the zip file!
21588: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
21589: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21590: - Menu: With <Crtl><F3> you can search for code on examples
21591: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21592: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
21593: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21594:
21595: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21596: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
21597: - Context Menu: You can printout your scripts as a pdf-file or html-export
21598: - Context: You do have a context menu with the right mouse click
21599:
21600: - Menu: With the UseCase Editor you can convert graphic formats too.
21601: - Menu: On menu Options you find Addons as compiled scripts
21602: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
21603: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21604: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21605: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
21606:         or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
21607:
21608: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21609: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21610: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21611: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
21612: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
21613: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
21614:          to delete and Click and mark to drag a bookmark
21615: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21616: - IDE: A file info with system and script information you find in menu Program/Information
21617: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21618: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21619: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21620: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21621: - Editor: Set Bookmarks to check your work in app or code
21622: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21623: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
21624: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21625:
21626: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
21627: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
21628: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
21629: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21630: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21631: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
21632: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21633: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21634: - IDE menu /Help/Tools/ open the Task Manager
21635:
21636: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21637: - Add on when no browser is available start /Options/Add ons/Easy Browser
21638: - Add on SOAP Tester with SOP POST File
21639: - Add on IP Protocol Sniffer with List View
21640: - Add on OpenGL mX Robot Demo for android
21641: - Add on Checkers Game, Add on Oscilloscope /View GEO Map View3
21642:
21643: - Menu: Help/Tools as a Tool Section with DOS Opener
21644: - Menu Editor: export the code as RTF File
21645: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21646: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21647: - Context: Auto Detect of Syntax depending on file extension
21648: - Code: some Windows API function start with w in the name like wGetAtomName();
21649: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21650: - IDE File Check with menu ..View/File Changes/...
21651: - Context: Create a Header with Create Header in Navigator List at right window
21652: - Code: use  SysErrorMessage to get a real Error Description, Ex.
21653:       RemoveDir('c:\NoSuchFolder') ;
21654:       writeln('System Error Message: '+ SysErrorMessage(GetLastError));
21655: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21656: - Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click
21657:
21658: - using DLL example in maXbox: //function: {*****************************************}
21659:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21660:                                  cb: DWORD): BOOL; //stdcall;;
21661:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
21662:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21663:      External  'OpenProcess@kernel32.dll stdcall';
```

```
21664:
21665: GCC Compile Ex Script
21666: procedure TFormMain_btnCompileClick(Sender: TObject);
21667: begin
21668:   AProcess:= TProcess.Create(Nil);
21669:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
21670:       + ' -o "' + OpenDialog2.FileName + '"';
21671:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21672:     AProcess.Execute;
21673:     Memo2.Lines.BeginUpdate;
21674:     Memo2.Lines.Clear;
21675:     Memo2.Lines.LoadFromStream(AProcess.Output);
21676:     Memo2.Lines.EndUpdate;
21677:   finally
21678:     AProcess.Free;
21679:   end;
21680: end;
21681:
21682: Stopwatch pattern
21683: Time1:= Time;
21684:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
21685:   if initAndStartBoard then
21686:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
21687:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
21688:   PrintF('%d %s',[Trunc((Time-Time1)*24),
21689:                   FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
21690:
21691:    POST git-receive-pack (chunked)
21692: Pushing to https://github.com/maxkleiner/maXbox3.git
21693: To https://github.com/maxkleiner/maXbox3.git f127d21..c6a98da  masterbox2 -> masterbox2
21694: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
21695:
21696: History Shell Hell
21697: PCT Precompile Technology , mX4 ScriptStudio
21698: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
21699: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21700: emax layers: system-package-component-unit-class-function-block
21701: new keywords def ref using maXCalcF
21702: UML: use case act class state seq pac comp dep - lib lab
21703: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21704: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21705: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21706: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21707: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21708: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21709: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21710: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
21711: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21712: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21713: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21714: Inno Install and Setup Routines
21715: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21716: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21717: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21718: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
21719: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
21720: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21721: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
21722: ByteCode2, IPUtils2, GEOCode, CGI-Powtils, GPS_2, External App
21723:
21724: Ref:
21725: https://unibe-ch.academia.edu/MaxKleiner
21726: http://www.slideshare.net/maxkleiner1
21727: http://www.scribd.com/max_kleiner
21728: http://www.delphiforfun.org/Programs/Utilities/index.htm
21729: http://www.slideshare.net/maxkleiner1
21730: http://s3.amazonaws.com/PreviewLinks/22959.html
21731: http://www.softwareschule.ch/arduino_training.pdf
21732: http://www.jrsoftware.org/isinfo.php
21733: http://www.be-precision.com/products/precision-builder/express/
21734: http://www.blaisepascal.eu/
21735: http://www.delphibasics.co.uk/
21736: http://www.youtube.com/watch?v=av89HAbqAsI
21737: http://www.angelfire.com/hi5/delphizeus/modal.html
21738: http://www.retroarchive.org/garbo/pc/turbopas/index.html
21739: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21740: http://delphi.org/2014/01/every-android-api-for-delphi/
21741: https://en.wikipedia.org/wiki/User:Maxkleiner
21742: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
21743: https://bitbucket.org/max_kleiner/maxbox3
21744: https://bitbucket.org/max_kleiner/maxbox3/downloads
21745:
21746:
21747: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
21748: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21749: =renderBasicSearchNarrative&q=%s';
21750: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
21751: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
21752:
```

```
21753:
21754: function OpenMap(const Data: string): boolean;
21755: var encURL: string;
21756: begin
21757:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
21758:   try
21759:    //HttpGet(EncodedURL, mapStream);   //WinInet
21760:    Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
21761:    //OpenDoc(Exepath+'openmapx.html');
21762:    S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
21763:   finally
21764:     encURL:= '';
21765:   end;
21766: end;
21767:
21768: procedure GetGEOMap(C_form,apath: string; const Data: string);
21769: var encodedURL: string;
21770:     mapStream: TMemoryStream;
21771: begin
21772:   //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPEncode(Data)]);
21773:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
21774:   mapStream:= TMemoryStream.create;
21775:   try
21776:     Wininet_HttpGet(EncodedURL, mapStream);   //WinInet
21777:     mapStream.Position:= 0;
21778:     mapStream.Savetofile(apath); // OpenDoc(apath);
21779:     S_ShellExecute(apath,'',seCmdOpen);
21780:   finally
21781:     mapStream.Free;
21782:   end;
21783: end;
21784:
21785:
21786:
21787:
21788: ************************************************************
21789:  unit List asm internal end
21790: ************************************************************
21791: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
21792: 02 unit SIRegister_IdStrings                   //Indy Sockets
21793: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
21794: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
21795: 05 unit IFSI_WinForm1puzzle;                   //maXbox
21796: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
21797: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
21798: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
21799: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
21800: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
21801: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
21802: 12 unit uPSCompiler.pas;                      //PS kernel functions
21803: 13 unit uPSI_DBCommon;                        //DB Common_Routines and Types
21804: 14 unit uPSI_Printers.pas                     //Delphi VCL
21805: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
21806: 16 unit uPSC_comobj;                         //COM Functions
21807: 17 unit uPSI_Clipbrd;                        //Delphi VCL
21808: 18 unit Filectrl in IFSI_SysUtils_max;       //VCL Runtime
21809: 19 unit uPSI_SqlExpr;                       //DBX3
21810: 20 unit uPSI_ADODB;                         //ADODB
21811: 21 unit uPSI_StrHlpr;                       //String Helper Routines
21812: 22 unit uPSI_DateUtils;                     //Expansion to DateTimeLib
21813: 23 unit uPSI_FileUtils;                     //Expansion to Sys/File Utils
21814: 24 unit JUtils / gsUtils;                   //Jedi / Metabase
21815: 25 unit JvFunctions_max;                    //Jedi Functions
21816: 26 unit HTTPParser;                         //Delphi VCL
21817: 27 unit HTTPUtil;                           //Delphi VCL
21818: 28 unit uPSI_XMLUtil;                       //Delphi VCL
21819: 29 unit uPSI_SOAPHTTPClient;                //Delphi VCL SOAP WebService V3.5
21820: 30 unit uPSI_Contnrs;                       //Delphi RTL Container of Classes
21821: 31 unit uPSI_MaskUtils;                     //RTL Edit and Mask functions
21822: 32 unit uPSI_MyBigInt;                      //big integer class with Math
21823: 33 unit uPSI_ConvUtils;                     //Delphi VCL Conversions engine
21824: 34 unit Types_Variants;                     //Delphi\Win32\rtl\sys
21825: 35 unit uPSI_IdHashSHA1;                    //Indy Crypto Lib
21826: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
21827: 37 unit uPSI_IdASN1Util;                    //Indy ASN1Utility Routines;
21828: 38 unit uPSI_IdLogFile;                     //Indy Logger from LogBase
21829: 39 unit uPSI_IdIcmpClient;                  //Indy Ping ICMP
21830: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
21831: 41 unit uPSI_FileCtrl;                      //Delphi RTL
21832: 42 unit uPSI_Outline;                       //Delphi VCL
21833: 43 unit uPSI_ScktComp;                      //Delphi RTL
21834: 44 unit uPSI_Calendar;                      //Delphi VCL
21835: 45 unit uPSI_VListView                      //VListView;
21836: 46 unit uPSI_DBGrids;                       //Delphi VCL
21837: 47 unit uPSI_DBCtrls;                       //Delphi VCL
21838: 48 unit ide_debugoutput;                    //maXbox
21839: 49 unit uPSI_ComCtrls;                      //Delphi VCL
21840: 50 unit uPSC_stdctrls+;                     //Delphi VCL
21841: 51 unit uPSI_Dialogs;                       //Delphi VCL
```

```
21842: 52 unit uPSI_StdConvs;                      //Delphi RTL
21843: 53 unit uPSI_DBClient;                      //Delphi RTL
21844: 54 unit uPSI_DBPlatform;                    //Delphi RTL
21845: 55 unit uPSI_Provider;                      //Delphi RTL
21846: 56 unit uPSI_FMTBcd;                        //Delphi RTL
21847: 57 unit uPSI_DBCGrids;                      //Delphi VCL
21848: 58 unit uPSI_CDSUtil;                       //MIDAS
21849: 59 unit uPSI_VarHlpr;                       //Delphi RTL
21850: 60 unit uPSI_ExtDlgs;                       //Delphi VCL
21851: 61 unit sdpStopwatch;                       //maXbox
21852: 62 unit uPSI_JclStatistics;                 //JCL
21853: 63 unit uPSI_JclLogic;                      //JCL
21854: 64 unit uPSI_JclMiscel;                     //JCL
21855: 65 unit uPSI_JclMath_max;                   //JCL RTL
21856: 66 unit uPSI_uTPLb_StreamUtils;             //LockBox 3
21857: 67 unit uPSI_MathUtils;                     //BCB
21858: 68 unit uPSI_JclMultimedia;                 //JCL
21859: 69 unit uPSI_WideStrUtils;                  //Delphi API/RTL
21860: 70 unit uPSI_GraphUtil;                     //Delphi RTL
21861: 71 unit uPSI_TypeTrans;                     //Delphi RTL
21862: 72 unit uPSI_HTTPApp;                       //Delphi VCL
21863: 73 unit uPSI_DBWeb;                         //Delphi VCL
21864: 74 unit uPSI_DBBdeWeb;                      //Delphi VCL
21865: 75 unit uPSI_DBXpressWeb;                   //Delphi VCL
21866: 76 unit uPSI_ShadowWnd;                     //Delphi VCL
21867: 77 unit uPSI_ToolWin;                       //Delphi VCL
21868: 78 unit uPSI_Tabs;                          //Delphi VCL
21869: 79 unit uPSI_JclGraphUtils;                 //JCL
21870: 80 unit uPSI_JclCounter;                    //JCL
21871: 81 unit uPSI_JclSysInfo;                    //JCL
21872: 82 unit uPSI_JclSecurity;                   //JCL
21873: 83 unit uPSI_JclFileUtils;                  //JCL
21874: 84 unit uPSI_IdUserAccounts;                //Indy
21875: 85 unit uPSI_IdAuthentication;              //Indy
21876: 86 unit uPSI_uTPLb_AES;                     //LockBox 3
21877: 87 unit uPSI_IdHashSHA1;                    //LockBox 3
21878: 88 unit uTPLb_BlockCipher;                  //LockBox 3
21879: 89 unit uPSI_ValEdit.pas;                   //Delphi VCL
21880: 90 unit uPSI_JvVCLUtils;                    //JCL
21881: 91 unit uPSI_JvDBUtil;                      //JCL
21882: 92 unit uPSI_JvDBUtils;                     //JCL
21883: 93 unit uPSI_JvAppUtils;                    //JCL
21884: 94 unit uPSI_JvCtrlUtils;                   //JCL
21885: 95 unit uPSI_JvFormToHtml;                  //JCL
21886: 96 unit uPSI_JvParsing;                     //JCL
21887: 97 unit uPSI_SerDlgs;                       //Toolbox
21888: 98 unit uPSI_Serial;                        //Toolbox
21889: 99 unit uPSI_JvComponent;                   //JCL
21890: 100 unit uPSI_JvCalc;                       //JCL
21891: 101 unit uPSI_JvBdeUtils;                   //JCL
21892: 102 unit uPSI_JvDateUtil;                   //JCL
21893: 103 unit uPSI_JvGenetic;                    //JCL
21894: 104 unit uPSI_JclBase;                      //JCL
21895: 105 unit uPSI_JvUtils;                      //JCL
21896: 106 unit uPSI_JvStrUtil;                    //JCL
21897: 107 unit uPSI_JvStrUtils;                   //JCL
21898: 108 unit uPSI_JvFileUtil;                   //JCL
21899: 109 unit uPSI_JvMemoryInfos;                //JCL
21900: 110 unit uPSI_JvComputerInfo;               //JCL
21901: 111 unit uPSI_JvgCommClasses;               //JCL
21902: 112 unit uPSI_JvgLogics;                    //JCL
21903: 113 unit uPSI_JvLED;                        //JCL
21904: 114 unit uPSI_JvTurtle;                     //JCL
21905: 115 unit uPSI_SortThds; unit uPSI_ThSort;   //maXbox
21906: 116 unit uPSI_JvgUtils;                     //JCL
21907: 117 unit uPSI_JvExprParser;                 //JCL
21908: 118 unit uPSI_HexDump;                      //Borland
21909: 119 unit uPSI_DBLogDlg;                     //VCL
21910: 120 unit uPSI_SqlTimSt;                     //RTL
21911: 121 unit uPSI_JvHtmlParser;                 //JCL
21912: 122 unit uPSI_JvgXMLSerializer;             //JCL
21913: 123 unit uPSI_JvJCLUtils;                   //JCL
21914: 124 unit uPSI_JvStrings;                    //JCL
21915: 125 unit uPSI_uTPLb_IntegerUtils;           //TurboPower
21916: 126 unit uPSI_uTPLb_HugeCardinal;           //TurboPower
21917: 127 unit uPSI_uTPLb_HugeCardinalUtils;      //TurboPower
21918: 128 unit uPSI_SynRegExpr;                   //SynEdit
21919: 129 unit uPSI_StUtils;                      //SysTools4
21920: 130 unit uPSI_StToHTML;                     //SysTools4
21921: 131 unit uPSI_StStrms;                      //SysTools4
21922: 132 unit uPSI_StFIN;                        //SysTools4
21923: 133 unit uPSI_StAstroP;                     //SysTools4
21924: 134 unit uPSI_StStat;                       //SysTools4
21925: 135 unit uPSI_StNetCon;                     //SysTools4
21926: 136 unit uPSI_StDecMth;                     //SysTools4
21927: 137 unit uPSI_StOStr;                       //SysTools4
21928: 138 unit uPSI_StPtrns;                      //SysTools4
21929: 139 unit uPSI_StNetMsg;                     //SysTools4
21930: 140 unit uPSI_StMath;                       //SysTools4
```

```
21931: 141 unit uPSI_StExpEng;                      //SysTools4
21932: 142 unit uPSI_StCRC;                         //SysTools4
21933: 143 unit uPSI_StExport,                      //SysTools4
21934: 144 unit uPSI_StExpLog,                      //SysTools4
21935: 145 unit uPSI_ActnList;                      //Delphi VCL
21936: 146 unit uPSI_jpeg;                          //Borland
21937: 147 unit uPSI_StRandom;                      //SysTools4
21938: 148 unit uPSI_StDict;                        //SysTools4
21939: 149 unit uPSI_StBCD;                         //SysTools4
21940: 150 unit uPSI_StTxtDat;                      //SysTools4
21941: 151 unit uPSI_StRegEx;                       //SysTools4
21942: 152 unit uPSI_IMouse;                        //VCL
21943: 153 unit uPSI_SyncObjs;                      //VCL
21944: 154 unit uPSI_AsyncCalls;                    //Hausladen
21945: 155 unit uPSI_ParallelJobs;                  //Saraiva
21946: 156 unit uPSI_Variants;                      //VCL
21947: 157 unit uPSI_VarCmplx;                      //VCL Wolfram
21948: 158 unit uPSI_DTDSchema;                     //VCL
21949: 159 unit uPSI_ShLwApi;                       //Brakel
21950: 160 unit uPSI_IBUtils;                       //VCL
21951: 161 unit uPSI_CheckLst;                      //VCL
21952: 162 unit uPSI_JvSimpleXml;                   //JCL
21953: 163 unit uPSI_JclSimpleXml;                  //JCL
21954: 164 unit uPSI_JvXmlDatabase;                 //JCL
21955: 165 unit uPSI_JvMaxPixel;                    //JCL
21956: 166 unit uPSI_JvItemsSearchs;                //JCL
21957: 167 unit uPSI_StExpEng2;                     //SysTools4
21958: 168 unit uPSI_StGenLog;                      //SysTools4
21959: 169 unit uPSI_JvLogFile;                     //Jcl
21960: 170 unit uPSI_CPort;                         //ComPort Lib v4.11
21961: 171 unit uPSI_CPortCtl;                      //ComPort
21962: 172 unit uPSI_CPortEsc;                      //ComPort
21963: 173 unit BarCodeScaner;                      //ComPort
21964: 174 unit uPSI_JvGraph;                       //JCL
21965: 175 unit uPSI_JvComCtrls;                    //JCL
21966: 176 unit uPSI_GUITesting;                    //D Unit
21967: 177 unit uPSI_JvFindFiles;                   //JCL
21968: 178 unit uPSI_StSystem;                      //SysTools4
21969: 179 unit uPSI_JvKeyboardStates;              //JCL
21970: 180 unit uPSI_JvMail;                        //JCL
21971: 181 unit uPSI_JclConsole;                    //JCL
21972: 182 unit uPSI_JclLANMan;                     //JCL
21973: 183 unit uPSI_IdCustomHTTPServer;            //Indy
21974: 184 unit IdHTTPServer                        //Indy
21975: 185 unit uPSI_IdTCPServer;                   //Indy
21976: 186 unit uPSI_IdSocketHandle;                //Indy
21977: 187 unit uPSI_IdIOHandlerSocket;             //Indy
21978: 188 unit IdIOHandler;                        //Indy
21979: 189 unit uPSI_cutils;                        //Bloodshed
21980: 190 unit uPSI_BoldUtils;                     //boldsoft
21981: 191 unit uPSI_IdSimpleServer;                //Indy
21982: 192 unit uPSI_IdSSLOpenSSL;                  //Indy
21983: 193 unit uPSI_IdMultipartFormData;           //Indy
21984: 194 unit uPSI_SynURIOpener;                  //SynEdit
21985: 195 unit uPSI_PerlRegEx;                     //PCRE
21986: 196 unit uPSI_IdHeaderList;                  //Indy
21987: 197 unit uPSI_StFirst;                       //SysTools4
21988: 198 unit uPSI_JvCtrls;                       //JCL
21989: 199 unit uPSI_IdTrivialFTPBase;              //Indy
21990: 200 unit uPSI_IdTrivialFTP;                  //Indy
21991: 201 unit uPSI_IdUDPBase;                     //Indy
21992: 202 unit uPSI_IdUDPClient;                   //Indy
21993: 203 unit uPSI_utypes;                        //for DMath.DLL
21994: 204 unit uPSI_ShellAPI;                      //Borland
21995: 205 unit uPSI_IdRemoteCMDClient;             //Indy
21996: 206 unit uPSI_IdRemoteCMDServer;             //Indy
21997: 207 unit IdRexecServer;                      //Indy
21998: 208 unit IdRexec; (unit uPSI_IdRexec;)       //Indy
21999: 209 unit IdUDPServer;                        //Indy
22000: 210 unit IdTimeUDPServer;                    //Indy
22001: 211 unit IdTimeServer;                      //Indy
22002: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
22003: 213 unit uPSI_IdIPWatch;                     //Indy
22004: 214 unit uPSI_IdIrcServer;                   //Indy
22005: 215 unit uPSI_IdMessageCollection;           //Indy
22006: 216 unit uPSI_cPEM;                          //Fundamentals 4
22007: 217 unit uPSI_cFundamentUtils;               //Fundamentals 4
22008: 218 unit uPSI_uwinplot;                      //DMath
22009: 219 unit uPSI_xrtl_util_CPUUtils;           //ExtentedRTL
22010: 220 unit uPSI_GR32_System;                   //Graphics32
22011: 221 unit uPSI_cFileUtils;                    //Fundamentals 4
22012: 222 unit uPSI_cDateTime; (timemachine)      //Fundamentals 4
22013: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
22014: 224 unit uPSI_cRandom;                       //Fundamentals 4
22015: 225 unit uPSI_ueval;                         //DMath
22016: 226 unit uPSI_xrtl_net_URIUtils;             //ExtendedRTL
22017: 227 unit xrtl_net_URIUtils;                  //ExtendedRTL
22018: 228 unit uPSI_ufft;  (FFT)                   //DMath
22019: 229 unit uPSI_DBXChannel;                    //Delphi
```

```
22020: 230 unit uPSI_DBXIndyChannel;                //Delphi Indy
22021: 231 unit uPSI_xrtl_util_COMCat;              //ExtendedRTL
22022: 232 unit uPSI_xrtl_util_StrUtils;            //ExtendedRTL
22023: 233 unit uPSI_xrtl_util_VariantUtils;        //ExtendedRTL
22024: 234 unit uPSI_xrtl_util_FileUtils;           //ExtendedRTL
22025: 235 unit xrtl_util_Compat;                   //ExtendedRTL
22026: 236 unit uPSI_OleAuto;                       //Borland
22027: 237 unit uPSI_xrtl_util_COMUtils;            //ExtendedRTL
22028: 238 unit uPSI_CmAdmCtl;                      //Borland
22029: 239 unit uPSI_ValEdit2;                      //VCL
22030: 240 unit uPSI_GR32;  //Graphics32           //Graphics32
22031: 241 unit uPSI_GR32_Image;                    //Graphics32
22032: 242 unit uPSI_xrtl_util_TimeUtils;           //ExtendedRTL
22033: 243 unit uPSI_xrtl_util_TimeZone;            //ExtendedRTL
22034: 244 unit uPSI_xrtl_util_TimeStamp;           //ExtendedRTL
22035: 245 unit uPSI_xrtl_util_Map;                 //ExtendedRTL
22036: 246 unit uPSI_xrtl_util_Set;                 //ExtendedRTL
22037: 247 unit uPSI_CPortMonitor;                  //ComPort
22038: 248 unit uPSI_StIniStm;                      //SysTools4
22039: 249 unit uPSI_GR32_ExtImage;                 //Graphics32
22040: 250 unit uPSI_GR32_OrdinalMaps;              //Graphics32
22041: 251 unit uPSI_GR32_Rasterizers;              //Graphics32
22042: 252 unit uPSI_xrtl_util_Exception;           //ExtendedRTL
22043: 253 unit uPSI_xrtl_util_Value;               //ExtendedRTL
22044: 254 unit uPSI_xrtl_util_Compare;             //ExtendedRTL
22045: 255 unit uPSI_FlatSB;                        //VCL
22046: 256 unit uPSI_JvAnalogClock;                 //JCL
22047: 257 unit uPSI_JvAlarms;                      //JCL
22048: 258 unit uPSI_JvSQLS;                        //JCL
22049: 259 unit uPSI_JvDBSecur;                     //JCL
22050: 260 unit uPSI_JvDBQBE;                       //JCL
22051: 261 unit uPSI_JvStarfield;                   //JCL
22052: 262 unit uPSI_JVCLMiscal;                    //JCL
22053: 263 unit uPSI_JvProfiler32;                  //JCL
22054: 264 unit uPSI_JvDirectories,                 //JCL
22055: 265 unit uPSI_JclSchedule,                   //JCL
22056: 266 unit uPSI_JclSvcCtrl,                    //JCL
22057: 267 unit uPSI_JvSoundControl,                //JCL
22058: 268 unit uPSI_JvBDESQLScript,                //JCL
22059: 269 unit uPSI_JvgDigits,                     //JCL>
22060: 270 unit uPSI_ImgList;                       //TCustomImageList
22061: 271 unit uPSI_JclMIDI;                       //JCL>
22062: 272 unit uPSI_JclWinMidi;                    //JCL>
22063: 273 unit uPSI_JclNTFS;                       //JCL>
22064: 274 unit uPSI_JclAppInst;                    //JCL>
22065: 275 unit uPSI_JvRle;                         //JCL>
22066: 276 unit uPSI_JvRas32;                       //JCL>
22067: 277 unit uPSI_JvImageDrawThread,             //JCL>
22068: 278 unit uPSI_JvImageWindow,                 //JCL>
22069: 279 unit uPSI_JvTransparentForm;             //JCL>
22070: 280 unit uPSI_JvWinDialogs;                  //JCL>
22071: 281 unit uPSI_JvSimLogic,                    //JCL>
22072: 282 unit uPSI_JvSimIndicator,                //JCL>
22073: 283 unit uPSI_JvSimPID,                      //JCL>
22074: 284 unit uPSI_JvSimPIDLinker,                //JCL>
22075: 285 unit uPSI_IdRFCReply;                    //Indy
22076: 286 unit uPSI_IdIdent;                       //Indy
22077: 287 unit uPSI_IdIdentServer;                 //Indy
22078: 288 unit uPSI_JvPatchFile;                   //JCL
22079: 289 unit uPSI_StNetPfm;                      //SysTools4
22080: 290 unit uPSI_StNet;                         //SysTools4
22081: 291 unit uPSI_JclPeImage;                    //JCL
22082: 292 unit uPSI_JclPrint;                      //JCL
22083: 293 unit uPSI_JclMime;                       //JCL
22084: 294 unit uPSI_JvRichEdit;                    //JCL
22085: 295 unit uPSI_JvDBRichEd;                    //JCL
22086: 296 unit uPSI_JvDice;                        //JCL
22087: 297 unit uPSI_JvFloatEdit;                   //JCL 3.9.8
22088: 298 unit uPSI_JvDirFrm;                      //JCL
22089: 299 unit uPSI_JvDualList;                    //JCL
22090: 300 unit uPSI_JvSwitch;                      ////JCL
22091: 301 unit uPSI_JvTimerLst;                    ////JCL
22092: 302 unit uPSI_JvMemTable;                    //JCL
22093: 303 unit uPSI_JvObjStr;                      //JCL
22094: 304 unit uPSI_StLArr;                        //SysTools4
22095: 305 unit uPSI_StWmDCpy;                      //SysTools4
22096: 306 unit uPSI_StText;                        //SysTools4
22097: 307 unit uPSI_StNTLog;                       //SysTools4
22098: 308 unit uPSI_xrtl_math_Integer;             //ExtendedRTL
22099: 309 unit uPSI_JvImagPrvw;                    //JCL
22100: 310 unit uPSI_JvFormPatch;                   //JCL
22101: 311 unit uPSI_JvPicClip;                     //JCL
22102: 312 unit uPSI_JvDataConv;                    //JCL
22103: 313 unit uPSI_JvCpuUsage;                    //JCL
22104: 314 unit uPSI_JclUnitConv_mX2;               //JCL
22105: 315 unit JvDualListForm;                     //JCL
22106: 316 unit uPSI_JvCpuUsage2;                   //JCL
22107: 317 unit uPSI_JvParserForm;                  //JCL
22108: 318 unit uPSI_JvJanTreeView;                 //JCL
```

```
22109: 319 unit uPSI_JvTransLED;                    //JCL
22110: 320 unit uPSI_JvPlaylist;                    //JCL
22111: 321 unit uPSI_JvFormAutoSize;                //JCL
22112: 322 unit uPSI_JvYearGridEditForm;            //JCL
22113: 323 unit uPSI_JvMarkupCommon;                //JCL
22114: 324 unit uPSI_JvChart;                       //JCL
22115: 325 unit uPSI_JvXPCore;                      //JCL
22116: 326 unit uPSI_JvXPCoreUtils;                 //JCL
22117: 327 unit uPSI_StatsClasses;                  //mX4
22118: 328 unit uPSI_ExtCtrls2;                     //VCL
22119: 329 unit uPSI_JvUrlGrabbers;                 //JCL
22120: 330 unit uPSI_JvXmlTree;                     //JCL
22121: 331 unit uPSI_JvWavePlayer;                  //JCL
22122: 332 unit uPSI_JvUnicodeCanvas;               //JCL
22123: 333 unit uPSI_JvTFUtils;                     //JCL
22124: 334 unit uPSI_IdServerIOHandler;             //Indy
22125: 335 unit uPSI_IdServerIOHandlerSocket;       //Indy
22126: 336 unit uPSI_IdMessageCoder;                //Indy
22127: 337 unit uPSI_IdMessageCoderMIME;            //Indy
22128: 338 unit uPSI_IdMIMETypes;                   //Indy
22129: 339 unit uPSI_JvConverter;                   //JCL
22130: 340 unit uPSI_JvCsvParse;                    //JCL
22131: 341 unit uPSI_umath;  unit uPSI_ugamma;      //DMath
22132: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
22133: 343 unit uPSI_JvDBGridExport;                //JCL
22134: 344 unit uPSI_JvgExport;                     //JCL
22135: 345 unit uPSI_JvSerialMaker;                 //JCL
22136: 346 unit uPSI_JvWin32;                       //JCL
22137: 347 unit uPSI_JvPaintFX;                     //JCL
22138: 348 unit uPSI_JvOracleDataSet; (beta)        //JCL
22139: 349 unit uPSI_JvValidators; (preview)        //JCL
22140: 350 unit uPSI_JvNTEventLog;                  //JCL
22141: 351 unit uPSI_ShellZipTool;                  //mX4
22142: 352 unit uPSI_JvJoystick;                    //JCL
22143: 353 unit uPSI_JvMailSlots;                   //JCL
22144: 354 unit uPSI_JclComplex;                    //JCL
22145: 355 unit uPSI_SynPdf;                        //Synopse
22146: 356 unit uPSI_Registry;                      //VCL
22147: 357 unit uPSI_TlHelp32;                      //VCL
22148: 358 unit uPSI_JclRegistry;                   //JCL
22149: 359 unit uPSI_JvAirBrush;                    //JCL
22150: 360 unit uPSI_mORMotReport;                  //Synopse
22151: 361 unit uPSI_JclLocales;                    //JCL
22152: 362 unit uPSI_SynEdit;                       //SynEdit
22153: 363 unit uPSI_SynEditTypes;                  //SynEdit
22154: 364 unit uPSI_SynMacroRecorder;              //SynEdit
22155: 365 unit uPSI_LongIntList;                   //SynEdit
22156: 366 unit uPSI_devcutils;                     //DevC
22157: 367 unit uPSI_SynEditMiscClasses;            //SynEdit
22158: 368 unit uPSI_SynEditRegexSearch;            //SynEdit
22159: 369 unit uPSI_SynEditHighlighter;            //SynEdit
22160: 370 unit uPSI_SynHighlighterPas;             //SynEdit
22161: 371 unit uPSI_JvSearchFiles;                 //JCL
22162: 372 unit uPSI_SynHighlighterAny;             //Lazarus
22163: 373 unit uPSI_SynEditKeyCmds;                //SynEdit
22164: 374 unit uPSI_SynEditMiscProcs,              //SynEdit
22165: 375 unit uPSI_SynEditKbdHandler              //SynEdit
22166: 376 unit uPSI_JvAppInst,                     //JCL
22167: 377 unit uPSI_JvAppEvent;                    //JCL
22168: 378 unit uPSI_JvAppCommand;                  //JCL
22169: 379 unit uPSI_JvAnimTitle;                   //JCL
22170: 380 unit uPSI_JvAnimatedImage;               //JCL
22171: 381 unit uPSI_SynEditExport;                 //SynEdit
22172: 382 unit uPSI_SynExportHTML;                 //SynEdit
22173: 383 unit uPSI_SynExportRTF;                  //SynEdit
22174: 384 unit uPSI_SynEditSearch;                 //SynEdit
22175: 385 unit uPSI_fMain_back                     //maXbox;
22176: 386 unit uPSI_JvZoom;                        //JCL
22177: 387 unit uPSI_PMrand;                        //PM
22178: 388 unit uPSI_JvSticker;                     //JCL
22179: 389 unit uPSI_XmlVerySimple;                 //mX4
22180: 390 unit uPSI_Services;                      //ExtPascal
22181: 391 unit uPSI_ExtPascalUtils;                //ExtPascal
22182: 392 unit uPSI_SocketsDelphi;                 //ExtPascal
22183: 393 unit uPSI_StBarC;                        //SysTools
22184: 394 unit uPSI_StDbBarC;                      //SysTools
22185: 395 unit uPSI_StBarPN;                       //SysTools
22186: 396 unit uPSI_StDbPNBC;                      //SysTools
22187: 397 unit uPSI_StDb2DBC;                      //SysTools
22188: 398 unit uPSI_StMoney;                       //SysTools
22189: 399 unit uPSI_JvForth;                       //JCL
22190: 400 unit uPSI_RestRequest;                   //mX4
22191: 401 unit uPSI_HttpRESTConnectionIndy;        //mX4
22192: 402 unit uPSI_JvXmlDatabase; //update       //JCL
22193: 403 unit uPSI_StAstro;                       //SysTools
22194: 404 unit uPSI_StSort;                        //SysTools
22195: 405 unit uPSI_StDate;                        //SysTools
22196: 406 unit uPSI_StDateSt;                      //SysTools
22197: 407 unit uPSI_StBase;                        //SysTools
```

```
22198: 408 unit uPSI_StVInfo;                        //SysTools
22199: 409 unit uPSI_JvBrowseFolder;                 //JCL
22200: 410 unit uPSI_JvBoxProcs;                     //JCL
22201: 411 unit uPSI_urandom; (unit uranuvag;)       //DMath
22202: 412 unit uPSI_usimann; (unit ugenalg;)        //DMath
22203: 413 unit uPSI_JvHighlighter;                  //JCL
22204: 414 unit uPSI_Diff;                          //mX4
22205: 415 unit uPSI_SpringWinAPI;                  //DSpring
22206: 416 unit uPSI_StBits;                        //SysTools
22207: 417 unit uPSI_TomDBQue;                      //mX4
22208: 418 unit uPSI_MultilangTranslator;           //mX4
22209: 419 unit uPSI_HyperLabel;                    //mX4
22210: 420 unit uPSI_Starter;                       //mX4
22211: 421 unit uPSI_FileAssocs;                    //devC
22212: 422 unit uPSI_devFileMonitorX;               //devC
22213: 423 unit uPSI_devrun;                        //devC
22214: 424 unit uPSI_devExec;                       //devC
22215: 425 unit uPSI_oysUtils;                      //devC
22216: 426 unit uPSI_DosCommand;                    //devC
22217: 427 unit uPSI_CppTokenizer;                  //devC
22218: 428 unit uPSI_JvHLParser;                    //devC
22219: 429 unit uPSI_JclMapi;                       //JCL
22220: 430 unit uPSI_JclShell;                      //JCL
22221: 431 unit uPSI_JclCOM;                        //JCL
22222: 432 unit uPSI_GR32_Math;                     //Graphics32
22223: 433 unit uPSI_GR32_LowLevel;                 //Graphics32
22224: 434 unit uPSI_SimpleHl;                      //mX4
22225: 435 unit uPSI_GR32_Filters,                  //Graphics32
22226: 436 unit uPSI_GR32_VectorMaps;               //Graphics32
22227: 437 unit uPSI_cXMLFunctions;                 //Fundamentals 4
22228: 438 unit uPSI_JvTimer;                       //JCL
22229: 439 unit uPSI_cHTTPUtils;                    //Fundamentals 4
22230: 440 unit uPSI_cTLSUtils;                     //Fundamentals 4
22231: 441 unit uPSI_JclGraphics;                   //JCL
22232: 442 unit uPSI_JclSynch;                      //JCL
22233: 443 unit uPSI_IdTelnet;                      //Indy
22234: 444 unit uPSI_IdTelnetServer,                //Indy
22235: 445 unit uPSI_IdEcho,                        //Indy
22236: 446 unit uPSI_IdEchoServer,                  //Indy
22237: 447 unit uPSI_IdEchoUDP,                     //Indy
22238: 448 unit uPSI_IdEchoUDPServer,               //Indy
22239: 449 unit uPSI_IdSocks,                       //Indy
22240: 450 unit uPSI_IdAntiFreezeBase;              //Indy
22241: 451 unit uPSI_IdHostnameServer;              //Indy
22242: 452 unit uPSI_IdTunnelCommon,                //Indy
22243: 453 unit uPSI_IdTunnelMaster,                //Indy
22244: 454 unit uPSI_IdTunnelSlave,                 //Indy
22245: 455 unit uPSI_IdRSH,                         //Indy
22246: 456 unit uPSI_IdRSHServer,                   //Indy
22247: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
22248: 458 unit uPSI_MapReader,                     //devC
22249: 459 unit uPSI_LibTar,                        //devC
22250: 460 unit uPSI_IdStack;                       //Indy
22251: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
22252: 462 unit uPSI_IdChargenServer;               //Indy
22253: 463 unit uPSI_IdFTPServer,                   //Indy
22254: 464 unit uPSI_IdException,                   //Indy
22255: 465 unit uPSI_utexplot;                      //DMath
22256: 466 unit uPSI_uwinstr;                       //DMath
22257: 467 unit uPSI_VarRecUtils;                   //devC
22258: 468 unit uPSI_JvStringListToHtml,            //JCL
22259: 469 unit uPSI_JvStringHolder,                //JCL
22260: 470 unit uPSI_IdCoder;                       //Indy
22261: 471 unit uPSI_SynHighlighterDfm;             //Synedit
22262: 472 unit uHighlighterProcs; in 471           //Synedit
22263: 473 unit uPSI_LazFileUtils,                  //LCL
22264: 474 unit uPSI_IDECmdLine;                    //LCL
22265: 475 unit uPSI_lazMasks;                      //LCL
22266: 476 unit uPSI_ip_misc;                       //mX4
22267: 477 unit uPSI_Barcode;                       //LCL
22268: 478 unit uPSI_SimpleXML;                     //LCL
22269: 479 unit uPSI_JclIniFiles;                   //JCL
22270: 480 unit uPSI_D2XXUnit;  {$X-}               //FTDI
22271: 481 unit uPSI_JclDateTime;                   //JCL
22272: 482 unit uPSI_JclEDI;                        //JCL
22273: 483 unit uPSI_JclMiscel2;                    //JCL
22274: 484 unit uPSI_JclValidation;                 //JCL
22275: 485 unit uPSI_JclAnsiStrings; {-PString}     //JCL
22276: 486 unit uPSI_SynEditMiscProcs2;             //Synedit
22277: 487 unit uPSI_JclStreams;                    //JCL
22278: 488 unit uPSI_QRCode;                        //mX4
22279: 489 unit uPSI_BlockSocket;                   //ExtPascal
22280: 490 unit uPSI_Masks,Utils                    //VCL
22281: 491 unit uPSI_synautil;                      //Synapse!
22282: 492 unit uPSI_JclMath_Class;                 //JCL RTL
22283: 493 unit ugamdist; //Gamma function         //DMath
22284: 494 unit uibeta, ucorrel; //IBeta           //DMath
22285: 495 unit uPSI_SRMgr;                         //mX4
22286: 496 unit uPSI_HotLog;                        //mX4
```

```
22287: 497 unit uPSI_DebugBox;                      //mX4
22288: 498 unit uPSI_ustrings;                      //DMath
22289: 499 unit uPSI_uregtest;                      //DMath
22290: 500 unit uPSI_usimplex;                      //DMath
22291: 501 unit uPSI_uhyper;                        //DMath
22292: 502 unit uPSI_IdHL7;                         //Indy
22293: 503 unit uPSI_IdIPMCastBase,                 //Indy
22294: 504 unit uPSI_IdIPMCastServer;               //Indy
22295: 505 unit uPSI_IdIPMCastClient;               //Indy
22296: 506 unit uPSI_unlfit; //nlregression        //DMath
22297: 507 unit uPSI_IdRawHeaders;                  //Indy
22298: 508 unit uPSI_IdRawClient;                   //Indy
22299: 509 unit uPSI_IdRawFunctions;                //Indy
22300: 510 unit uPSI_IdTCPStream;                   //Indy
22301: 511 unit uPSI_IdSNPP;                        //Indy
22302: 512 unit uPSI_St2DBarC;                      //SysTools
22303: 513 unit uPSI_ImageWin;  //FTL              //VCL
22304: 514 unit uPSI_CustomDrawTreeView; //FTL     //VCL
22305: 515 unit uPSI_GraphWin;  //FTL              //VCL
22306: 516 unit uPSI_actionMain;  //FTL            //VCL
22307: 517 unit uPSI_StSpawn;                       //SysTools
22308: 518 unit uPSI_CtlPanel;                      //VCL
22309: 519 unit uPSI_IdLPR;                         //Indy
22310: 520 unit uPSI_SockRequestInterpreter;        //Indy
22311: 521 unit uPSI_ulambert;                      //DMath
22312: 522 unit uPSI_ucholesk;                      //DMath
22313: 523 unit uPSI_SimpleDS;                      //VCL
22314: 524 unit uPSI_DBXSqlScanner;                 //VCL
22315: 525 unit uPSI_DBXMetaDataUtil;               //VCL
22316: 526 unit uPSI_Chart;                         //TEE
22317: 527 unit uPSI_TeeProcs;                      //TEE
22318: 528 unit mXBDEUtils;                         //mX4
22319: 529 unit uPSI_MDIEdit;                       //VCL
22320: 530 unit uPSI_CopyPrsr;                      //VCL
22321: 531 unit uPSI_SockApp;                       //VCL
22322: 532 unit uPSI_AppEvnts;                      //VCL
22323: 533 unit uPSI_ExtActns;                      //VCL
22324: 534 unit uPSI_TeEngine;                      //TEE
22325: 535 unit uPSI_CoolMain; //browser           //VCL
22326: 536 unit uPSI_StCRC;                         //SysTools
22327: 537 unit uPSI_StDecMth2;                     //SysTools
22328: 538 unit uPSI_frmExportMain;                 //Synedit
22329: 539 unit uPSI_SynDBEdit;                     //Synedit
22330: 540 unit uPSI_SynEditWildcardSearch;         //Synedit
22331: 541 unit uPSI_BoldComUtils;                  //BOLD
22332: 542 unit uPSI_BoldIsoDateTime;               //BOLD
22333: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils   //BOLD
22334: 544 unit uPSI_BoldXMLRequests;               //BOLD
22335: 545 unit uPSI_BoldStringList;                //BOLD
22336: 546 unit uPSI_BoldFileHandler;               //BOLD
22337: 547 unit uPSI_BoldContainers;                //BOLD
22338: 548 unit uPSI_BoldQueryUserDlg;              //BOLD
22339: 549 unit uPSI_BoldWinINet;                   //BOLD
22340: 550 unit uPSI_BoldQueue;                     //BOLD
22341: 551 unit uPSI_JvPcx;                         //JCL
22342: 552 unit uPSI_IdWhois;                       //Indy
22343: 553 unit uPSI_IdWhoIsServer;                 //Indy
22344: 554 unit uPSI_IdGopher;                      //Indy
22345: 555 unit uPSI_IdDateTimeStamp;               //Indy
22346: 556 unit uPSI_IdDayTimeServer;               //Indy
22347: 557 unit uPSI_IdDayTimeUDP;                  //Indy
22348: 558 unit uPSI_IdDayTimeUDPServer;            //Indy
22349: 559 unit uPSI_IdDICTServer;                  //Indy
22350: 560 unit uPSI_IdDiscardServer;               //Indy
22351: 561 unit uPSI_IdDiscardUDPServer;            //Indy
22352: 562 unit uPSI_IdMappedFTP;                   //Indy
22353: 563 unit uPSI_IdMappedPortTCP;               //Indy
22354: 564 unit uPSI_IdGopherServer;                //Indy
22355: 565 unit uPSI_IdQotdServer;                  //Indy
22356: 566 unit uPSI_JvRgbToHtml;                   //JCL
22357: 567 unit uPSI_JvRemLog,                      //JCL
22358: 568 unit uPSI_JvSysComp;                     //JCL
22359: 569 unit uPSI_JvTMTL;                        //JCL
22360: 570 unit uPSI_JvWinampAPI;                   //JCL
22361: 571 unit uPSI_MSysUtils;                     //mX4
22362: 572 unit uPSI_ESBMaths;                      //ESB
22363: 573 unit uPSI_ESBMaths2;                     //ESB
22364: 574 unit uPSI_uLkJSON;                       //Lk
22365: 575 unit uPSI_ZURL;  //Zeos                  //Zeos
22366: 576 unit uPSI_ZSysUtils;                     //Zeos
22367: 577 unit unaUtils internals                  //UNA
22368: 578 unit uPSI_ZMatchPattern;                 //Zeos
22369: 579 unit uPSI_ZClasses;                      //Zeos
22370: 580 unit uPSI_ZCollections;                  //Zeos
22371: 581 unit uPSI_ZEncoding;                     //Zeos
22372: 582 unit uPSI_IdRawBase;                     //Indy
22373: 583 unit uPSI_IdNTLM;                        //Indy
22374: 584 unit uPSI_IdNNTP;                        //Indy
22375: 585 unit uPSI_usniffer; //PortScanForm       //mX4
```

```
22376: 586 unit uPSI_IdCoderMIME;                    //Indy
22377: 587 unit uPSI_IdCoderUUE;                     //Indy
22378: 588 unit uPSI_IdCoderXXE;                     //Indy
22379: 589 unit uPSI_IdCoder3to4;                    //Indy
22380: 590 unit uPSI_IdCookie;                       //Indy
22381: 591 unit uPSI_IdCookieManager;                //Indy
22382: 592 unit uPSI_WDosSocketUtils;                //WDos
22383: 593 unit uPSI_WDosPlcUtils;                   //WDos
22384: 594 unit uPSI_WDosPorts;                      //WDos
22385: 595 unit uPSI_WDosResolvers;                  //WDos
22386: 596 unit uPSI_WDosTimers;                     //WDos
22387: 597 unit uPSI_WDosPlcs;                       //WDos
22388: 598 unit uPSI_WDosPneumatics;                 //WDos
22389: 599 unit uPSI_IdFingerServer;                 //Indy
22390: 600 unit uPSI_IdDNSResolver;                  //Indy
22391: 601 unit uPSI_IdHTTPWebBrokerBridge;          //Indy
22392: 602 unit uPSI_IdIntercept;                    //Indy
22393: 603 unit uPSI_IdIPMCastBase;                  //Indy
22394: 604 unit uPSI_IdLogBase;                      //Indy
22395: 605 unit uPSI_IdIOHandlerStream;              //Indy
22396: 606 unit uPSI_IdMappedPortUDP;                //Indy
22397: 607 unit uPSI_IdQOTDUDPServer;                //Indy
22398: 608 unit uPSI_IdQOTDUDP;                      //Indy
22399: 609 unit uPSI_IdSysLog;                       //Indy
22400: 610 unit uPSI_IdSysLogServer;                 //Indy
22401: 611 unit uPSI_IdSysLogMessage;                //Indy
22402: 612 unit uPSI_IdTimeServer;                   //Indy
22403: 613 unit uPSI_IdTimeUDP;                      //Indy
22404: 614 unit uPSI_IdTimeUDPServer;                //Indy
22405: 615 unit uPSI_IdUserAccounts;                 //Indy
22406: 616 unit uPSI_TextUtils;                      //mX4
22407: 617 unit uPSI_MandelbrotEngine;               //mX4
22408: 618 unit uPSI_delphi_arduino_Unit1;           //mX4
22409: 619 unit uPSI_DTDSchema2;                     //mX4
22410: 620 unit uPSI_fplotMain;                      //DMath
22411: 621 unit uPSI_FindFileIter;                   //mX4
22412: 622 unit uPSI_PppState;  (JclStrHashMap)      //PPP
22413: 623 unit uPSI_PppParser;                      //PPP
22414: 624 unit uPSI_PppLexer;                       //PPP
22415: 625 unit uPSI_PCharUtils;                     //PPP
22416: 626 unit uPSI_uJSON;                          //WU
22417: 627 unit uPSI_JclStrHashMap;                  //JCL
22418: 628 unit uPSI_JclHookExcept;                  //JCL
22419: 629 unit uPSI_EncdDecd;                       //VCL
22420: 630 unit uPSI_SockAppReg;                     //VCL
22421: 631 unit uPSI_PJFileHandle;                   //PJ
22422: 632 unit uPSI_PJEnvVars;                      //PJ
22423: 633 unit uPSI_PJPipe;                         //PJ
22424: 634 unit uPSI_PJPipeFilters;                  //PJ
22425: 635 unit uPSI_PJConsoleApp;                   //PJ
22426: 636 unit uPSI_UConsoleAppEx;                  //PJ
22427: 637 unit uPSI_DbxSocketChannelNative,         //VCL
22428: 638 unit uPSI_DbxDataGenerator,               //VCL
22429: 639 unit uPSI_DBXClient;                      //VCL
22430: 640 unit uPSI_IdLogEvent;                     //Indy
22431: 641 unit uPSI_Reversi;                        //mX4
22432: 642 unit uPSI_Geometry;                       //mX4
22433: 643 unit uPSI_IdSMTPServer;                   //Indy
22434: 644 unit uPSI_Textures;                       //mX4
22435: 645 unit uPSI_IBX;                            //VCL
22436: 646 unit uPSI_IWDBCommon;                     //VCL
22437: 647 unit uPSI_SortGrid;                       //mX4
22438: 648 unit uPSI_IB;                             //VCL
22439: 649 unit uPSI_IBScript;                       //VCL
22440: 650 unit uPSI_JvCSVBaseControls;              //JCL
22441: 651 unit uPSI_Jvg3DColors;                    //JCL
22442: 652 unit uPSI_JvHLEditor;  //beat             //JCL
22443: 653 unit uPSI_JvShellHook;                    //JCL
22444: 654 unit uPSI_DBCommon2                       //VCL
22445: 655 unit uPSI_JvSHFileOperation;              //JCL
22446: 656 unit uPSI_uFilexport;                     //mX4
22447: 657 unit uPSI_JvDialogs;                      //JCL
22448: 658 unit uPSI_JvDBTreeView;                   //JCL
22449: 659 unit uPSI_JvDBUltimGrid;                  //JCL
22450: 660 unit uPSI_JvDBQueryParamsForm;            //JCL
22451: 661 unit uPSI_JvExControls;                   //JCL
22452: 662 unit uPSI_JvBDEMemTable;                  //JCL
22453: 663 unit uPSI_JvCommStatus;                   //JCL
22454: 664 unit uPSI_JvMailSlots2;                   //JCL
22455: 665 unit uPSI_JvgWinMask;                     //JCL
22456: 666 unit uPSI_StEclpse;                       //SysTools
22457: 667 unit uPSI_StMime;                         //SysTools
22458: 668 unit uPSI_StList;                         //SysTools
22459: 669 unit uPSI_StMerge;                        //SysTools
22460: 670 unit uPSI_StStrS;                         //SysTools
22461: 671 unit uPSI_StTree,                         //SysTools
22462: 672 unit uPSI_StVArr;                         //SysTools
22463: 673 unit uPSI_StRegIni;                       //SysTools
22464: 674 unit uPSI_urkf;                           //DMath
```

```
22465: 675 unit uPSI_usvd;                          //DMath
22466: 676 unit uPSI_DepWalkUtils;                  //JCL
22467: 677 unit uPSI_OptionsFrm;                    //JCL
22468: 678 unit yuvconverts;                       //mX4
22469: 679 uPSI_JvPropAutoSave;                    //JCL
22470: 680 uPSI_AclAPI;                            //alcinoe
22471: 681 uPSI_AviCap;                            //alcinoe
22472: 682 uPSI_ALAVLBinaryTree;                   //alcinoe
22473: 683 uPSI_ALFcnMisc;                         //alcinoe
22474: 684 uPSI_ALStringList;                      //alcinoe
22475: 685 uPSI_ALQuickSortList;                   //alcinoe
22476: 686 uPSI_ALStaticText;                      //alcinoe
22477: 687 uPSI_ALJSONDoc;                         //alcinoe
22478: 688 uPSI_ALGSMComm;                         //alcinoe
22479: 689 uPSI_ALWindows;                         //alcinoe
22480: 690 uPSI_ALMultiPartFormDataParser;         //alcinoe
22481: 691 uPSI_ALHttpCommon;                      //alcinoe
22482: 692 uPSI_ALWebSpider,                       //alcinoe
22483: 693 uPSI_ALHttpClient;                      //alcinoe
22484: 694 uPSI_ALFcnHTML;                         //alcinoe
22485: 695 uPSI_ALFTPClient;                       //alcinoe
22486: 696 uPSI_ALInternetMessageCommon;           //alcinoe
22487: 697 uPSI_ALWininetHttpClient;               //alcinoe
22488: 698 uPSI_ALWinInetFTPClient;                //alcinoe
22489: 699 uPSI_ALWinHttpWrapper;                  //alcinoe
22490: 700 uPSI_ALWinHttpClient;                   //alcinoe
22491: 701 uPSI_ALFcnWinSock;                      //alcinoe
22492: 702 uPSI_ALFcnSQL;                          //alcinoe
22493: 703 uPSI_ALFcnCGI;                          //alcinoe
22494: 704 uPSI_ALFcnExecute;                      //alcinoe
22495: 705 uPSI_ALFcnFile;                         //alcinoe
22496: 706 uPSI_ALFcnMime;                         //alcinoe
22497: 707 uPSI_ALPhpRunner;                       //alcinoe
22498: 708 uPSI_ALGraphic;                         //alcinoe
22499: 709 uPSI_ALIniFiles;                        //alcinoe
22500: 710 uPSI_ALMemCachedClient;                 //alcinoe
22501: 711 unit uPSI_MyGrids;                      //mX4
22502: 712 uPSI_ALMultiPartMixedParser             //alcinoe
22503: 713 uPSI_ALSMTPClient                       //alcinoe
22504: 714 uPSI_ALNNTPClient;                      //alcinoe
22505: 715 uPSI_ALHintBalloon;                     //alcinoe
22506: 716 unit uPSI_ALXmlDoc;                     //alcinoe
22507: 717 unit uPSI_IPCThrd;                      //VCL
22508: 718 unit uPSI_MonForm;                      //VCL
22509: 719 unit uPSI_TeCanvas;                     //Orpheus
22510: 720 unit uPSI_Ovcmisc;                      //Orpheus
22511: 721 unit uPSI_ovcfiler;                     //Orpheus
22512: 722 unit uPSI_ovcstate;                     //Orpheus
22513: 723 unit uPSI_ovccoco;                      //Orpheus
22514: 724 unit uPSI_ovcrvexp;                     //Orpheus
22515: 725 unit uPSI_OvcFormatSettings;            //Orpheus
22516: 726 unit uPSI_OvcUtils;                     //Orpheus
22517: 727 unit uPSI_ovcstore;                     //Orpheus
22518: 728 unit uPSI_ovcstr;                       //Orpheus
22519: 729 unit uPSI_ovcmru;                       //Orpheus
22520: 730 unit uPSI_ovccmd;                       //Orpheus
22521: 731 unit uPSI_ovctimer;                     //Orpheus
22522: 732 unit uPSI_ovcintl;                      //Orpheus
22523: 733 uPSI_AfCircularBuffer;                  //AsyncFree
22524: 734 uPSI_AfUtils;                           //AsyncFree
22525: 735 uPSI_AfSafeSync;                        //AsyncFree
22526: 736 uPSI_AfComPortCore;                     //AsyncFree
22527: 737 uPSI_AfComPort;                         //AsyncFree
22528: 738 uPSI_AfPortControls;                    //AsyncFree
22529: 739 uPSI_AfDataDispatcher;                  //AsyncFree
22530: 740 uPSI_AfViewers;                         //AsyncFree
22531: 741 uPSI_AfDataTerminal;                    //AsyncFree
22532: 742 uPSI_SimplePortMain;                    //AsyncFree
22533: 743 unit uPSI_ovcclock;                     //Orpheus
22534: 744 unit uPSI_o32intlst;                    //Orpheus
22535: 745 unit uPSI_o32ledlabel;                  //Orpheus
22536: 746 unit uPSI_AlMySqlClient;                //alcinoe
22537: 747 unit uPSI_ALFBXClient;                  //alcinoe
22538: 748 unit uPSI_ALFcnSQL;                     //alcinoe
22539: 749 unit uPSI_AsyncTimer;                   //mX4
22540: 750 unit uPSI_ApplicationFileIO;            //mX4
22541: 751 unit uPSI_PsAPI;                        //VCLé
22542: 752 uPSI_ovcuser;                           //Orpheus
22543: 753 uPSI_ovcurl;                            //Orpheus
22544: 754 uPSI_ovcvlb;                            //Orpheus
22545: 755 uPSI_ovccolor;                          //Orpheus
22546: 756 uPSI_ALFBXLib,                          //alcinoe
22547: 757 uPSI_ovcmeter;                          //Orpheus
22548: 758 uPSI_ovcpeakm;                          //Orpheus
22549: 759 uPSI_O32BGSty;                          //Orpheus
22550: 760 uPSI_ovcBidi;                           //Orpheus
22551: 761 uPSI_ovctcary;                          //Orpheus
22552: 762 uPSI_DXPUtils;                          //mX4
22553: 763 uPSI_ALMultiPartBaseParser;             //alcinoe
```

```
22554: 764 uPSI_ALMultiPartAlternativeParser;         //alcinoe
22555: 765 uPSI_ALPOP3Client;                        //alcinoe
22556: 766 uPSI_SmallUtils;                          //mX4
22557: 767 uPSI_MakeApp;                             //mX4
22558: 768 uPSI_O32MouseMon;                         //Orpheus
22559: 769 uPSI_OvcCache;                           //Orpheus
22560: 770 uPSI_ovccalc;                            //Orpheus
22561: 771 uPSI_Joystick,                           //OpenGL
22562: 772 uPSI_ScreenSaver;                        //OpenGL
22563: 773 uPSI_XCollection,                        //OpenGL
22564: 774 uPSI_Polynomials,                        //OpenGL
22565: 775 uPSI_PersistentClasses, //9.86          //OpenGL
22566: 776 uPSI_VectorLists;                        //OpenGL
22567: 777 uPSI_XOpenGL,                           //OpenGL
22568: 778 uPSI_MeshUtils;                         //OpenGL
22569: 779 unit uPSI_JclSysUtils;                  //JCL
22570: 780 unit uPSI_JclBorlandTools;              //JCL
22571: 781 unit JclFileUtils_max;                  //JCL
22572: 782 uPSI_AfDataControls,                    //AsyncFree
22573: 783 uPSI_GLSilhouette;                      //OpenGL
22574: 784 uPSI_JclSysUtils_class;                 //JCL
22575: 785 uPSI_JclFileUtils_class;                //JCL
22576: 786 uPSI_FileUtil;                          //JCL
22577: 787 uPSI_changefind;                        //mX4
22578: 788 uPSI_cmdIntf;                           //mX4
22579: 789 uPSI_fservice;                          //mX4
22580: 790 uPSI_Keyboard;                          //OpenGL
22581: 791 uPSI_VRMLParser,                        //OpenGL
22582: 792 uPSI_GLFileVRML,                        //OpenGL
22583: 793 uPSI_Octree;                            //OpenGL
22584: 794 uPSI_GLPolyhedron,                      //OpenGL
22585: 795 uPSI_GLCrossPlatform;                   //OpenGL
22586: 796 uPSI_GLParticles;                       //OpenGL
22587: 797 uPSI_GLNavigator;                       //OpenGL
22588: 798 uPSI_GLStarRecord;                      //OpenGL
22589: 799 uPSI_GLTextureCombiners;                //OpenGL
22590: 800 uPSI_GLCanvas;                          //OpenGL
22591: 801 uPSI_GeometryBB;                        //OpenGL
22592: 802 uPSI_GeometryCoordinates;               //OpenGL
22593: 803 uPSI_VectorGeometry;                    //OpenGL
22594: 804 uPSI_BumpMapping;                       //OpenGL
22595: 805 uPSI_TGA;                               //OpenGL
22596: 806 uPSI_GLVectorFileObjects;               //OpenGL
22597: 807 uPSI_IMM;                               //VCL
22598: 808 uPSI_CategoryButtons;                   //VCL
22599: 809 uPSI_ButtonGroup;                       //VCL
22600: 810 uPSI_DbExcept;                          //VCL
22601: 811 uPSI_AxCtrls;                           //VCL
22602: 812 uPSI_GL_actorUnit1;                     //OpenGL
22603: 813 uPSI_StdVCL;                            //VCL
22604: 814 unit CurvesAndSurfaces;                 //OpenGL
22605: 815 uPSI_DataAwareMain;                     //AsyncFree
22606: 816 uPSI_TabNotBk;                          //VCL
22607: 817 uPSI_udwsfiler;                         //mX4
22608: 818 uPSI_synaip;                            //Synapse!
22609: 819 uPSI_synacode;                          //Synapse
22610: 820 uPSI_synachar;                          //Synapse
22611: 821 uPSI_synamisc;                          //Synapse
22612: 822 uPSI_synaser;                           //Synapse
22613: 823 uPSI_synaicnv;                          //Synapse
22614: 824 uPSI_tlntsend;                          //Synapse
22615: 825 uPSI_pingsend;                          //Synapse
22616: 826 uPSI_blcksock;                          //Synapse
22617: 827 uPSI_asn1util;                          //Synapse
22618: 828 uPSI_dnssend;                           //Synapse
22619: 829 uPSI_clamsend;                          //Synapse
22620: 830 uPSI_ldapsend;                          //Synapse
22621: 831 uPSI_mimemess;                          //Synapse
22622: 832 uPSI_slogsend;                          //Synapse
22623: 833 uPSI_mimepart;                          //Synapse
22624: 834 uPSI_mimeinln;                          //Synapse
22625: 835 uPSI_ftpsend,                           //Synapse
22626: 836 uPSI_ftptsend;                          //Synapse
22627: 837 uPSI_httpsend;                          //Synapse
22628: 838 uPSI_sntpsend;                          //Synapse
22629: 839 uPSI_smtpsend;                          //Synapse
22630: 840 uPSI_snmpsend;                          //Synapse
22631: 841 uPSI_imapsend;                          //Synapse
22632: 842 uPSI_pop3send;                          //Synapse
22633: 843 uPSI_nntpsend;                          //Synapse
22634: 844 uPSI_ssl_cryptlib;                      //Synapse
22635: 845 uPSI_ssl_openssl;                       //Synapse
22636: 846 uPSI_synhttp_daemon;                    //Synapse
22637: 847 uPSI_NetWork;                           //mX4
22638: 848 uPSI_PingThread;                        //Synapse
22639: 849 uPSI_JvThreadTimer;                     //JCL
22640: 850 unit uPSI_wwSystem;                     //InfoPower
22641: 851 unit uPSI_IdComponent;                  //Indy
22642: 852 unit uPSI_IdIOHandlerThrottle;          //Indy
```

```
22643: 853 unit uPSI_Themes;                          //VCL
22644: 854 unit uPSI_StdStyleActnCtrls;               //VCL
22645: 855 unit uPSI_UDDIHelper;                      //VCL
22646: 856 unit uPSI_IdIMAP4Server;                   //Indy
22647: 857 uPSI_VariantSymbolTable,                   //VCL //3.9.9.92
22648: 858 uPSI_udf_glob,                             //mX4
22649: 859 uPSI_TabGrid,                              //VCL
22650: 860 uPSI_JsDBTreeView,                         //mX4
22651: 861 uPSI_JsSendMail,                           //mX4
22652: 862 uPSI_dbTvRecordList,                       //mX4
22653: 863 uPSI_TreeVwEx,                            //mX4
22654: 864 uPSI_ECDataLink,                          //mX4
22655: 865 uPSI_dbTree,                              //mX4
22656: 866 uPSI_dbTreeCBox,                          //mX4
22657: 867 unit uPSI_Debug; //TfrmDebug             //mX4
22658: 868 uPSI_TimeFncs;                            //mX4
22659: 869 uPSI_FileIntf,                            //VCL
22660: 870 uPSI_SockTransport,                       //RTL
22661: 871 unit uPSI_WinInet;                        //RTL
22662: 872 unit uPSI_Wwstr;                          //mX4
22663: 873 uPSI_DBLookup,                            //VCL
22664: 874 uPSI_Hotspot,                             //mX4
22665: 875 uPSI_HList; //History List                //mX4
22666: 876 unit uPSI_DrTable;                        //VCL
22667: 877 uPSI_TConnect,                            //VCL
22668: 878 uPSI_DataBkr,                             //VCL
22669: 879 uPSI_HTTPIntr;                            //VCL
22670: 880 unit uPSI_Mathbox;                        //mX4
22671: 881 uPSI_cyIndy,                              //cY
22672: 882 uPSI_cySysUtils,                          //cY
22673: 883 uPSI_cyWinUtils,                          //cY
22674: 884 uPSI_cyStrUtils,                          //cY
22675: 885 uPSI_cyObjUtils,                          //cY
22676: 886 uPSI_cyDateUtils,                         //cY
22677: 887 uPSI_cyBDE,                               //cY
22678: 888 uPSI_cyClasses,                           //cY
22679: 889 uPSI_cyGraphics,  //3.9.9.94_2           //cY
22680: 890 unit uPSI_cyTypes;                        //cY
22681: 891 uPSI_JvDateTimePicker,                    //JCL
22682: 892 uPSI_JvCreateProcess,                     //JCL
22683: 893 uPSI_JvEasterEgg,                         //JCL
22684: 894 uPSI_WinSvc,  //3.9.9.94_3               //VCL
22685: 895 uPSI_SvcMgr                              //VCL
22686: 896 unit uPSI_JvPickDate;                     //JCL
22687: 897 unit uPSI_JvNotify;                      //JCL
22688: 898 uPSI_JvStrHlder                          //JCL
22689: 899 unit uPSI_JclNTFS2;                      //JCL
22690: 900 uPSI_Jcl8087 //math coprocessor          //JCL
22691: 901 uPSI_JvAddPrinter                        //JCL
22692: 902 uPSI_JvCabFile                           //JCL
22693: 903 uPSI_JvDataEmbedded;                     //JCL
22694: 904 unit uPSI_U_HexView;                     //mX4
22695: 905 uPSI_UWavein4,                           //mX4
22696: 906 uPSI_AMixer,                             //mX4
22697: 907 uPSI_JvaScrollText,                      //mX4
22698: 908 uPSI_JvArrow,                            //mX4
22699: 909 unit uPSI_UrlMon;                        //mX4
22700: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
22701: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain; //DFF
22702: 912 unit uPSI_DFFUtils;                      //DFF
22703: 913 unit uPSI_MathsLib;                      //DFF
22704: 914 uPSI_UIntList;                           //DFF
22705: 915 uPSI_UGetParens;                         //DFF
22706: 916 unit uPSI_UGeometry;                     //DFF
22707: 917 unit uPSI_UAstronomy;                    //DFF
22708: 918 unit uPSI_UCardComponentV2;              //DFF
22709: 919 unit uPSI_UTGraphSearch;                 //DFF
22710: 920 unit uPSI_UParser10;                     //DFF
22711: 921 unit uPSI_cyIEUtils;                     //cY
22712: 922 unit uPSI_UcomboV2;                      //DFF
22713: 923 uPSI_cyBaseComm,                         //cY
22714: 924 uPSI_cyAppInstances,                     //cY
22715: 925 uPSI_cyAttract,                          //cY
22716: 926 uPSI_cyDERUtils                          //cY
22717: 927 unit uPSI_cyDocER;                       //cY
22718: 928 unit uPSI_ODBC;                          //mX
22719: 929 unit uPSI_AssocExec;                     //mX
22720: 930 uPSI_cyBaseCommRoomConnector,            //cY
22721: 931 uPSI_cyCommRoomConnector,                //cY
22722: 932 uPSI_cyCommunicate,                      //cY
22723: 933 uPSI_cyImage;                            //cY
22724: 934 uPSI_cyBaseContainer                     //cY
22725: 935 uPSI_cyModalContainer,                   //cY
22726: 936 uPSI_cyFlyingContainer;                  //cY
22727: 937 uPSI_RegStr,                             //VCL
22728: 938 uPSI_HtmlHelpViewer;                     //VCL
22729: 939 unit uPSI_cyIniForm                      //cY
22730: 940 unit uPSI_cyVirtualGrid;                 //cY
22731: 941 uPSI_Profiler,                           //DA
```

```
22732: 942 uPSI_BackgroundWorker,                    //DA
22733: 943 uPSI_WavePlay,                            //DA
22734: 944 uPSI_WaveTimer,                           //DA
22735: 945 uPSI_WaveUtils;                           //DA
22736: 946 uPSI_NamedPipes,                         //TB
22737: 947 uPSI_NamedPipeServer,                    //TB
22738: 948 unit uPSI_process,                       //TB
22739: 949 unit uPSI_DPUtils;                       //TB
22740: 950 unit uPSI_CommonTools;                   //TB
22741: 951 uPSI_DataSendToWeb,                      //TB
22742: 952 uPSI_StarCalc,                           //TB
22743: 953 uPSI_D2_XPVistaHelperU                   //TB
22744: 954 unit uPSI_NetTools                       //TB
22745: 955 unit uPSI_Pipes                          //TB
22746: 956 uPSI_ProcessUnit,                       //mX
22747: 957 uPSI_adGSM,                             //mX
22748: 958 unit uPSI_BetterADODataSet;             //mX
22749: 959 unit uPSI_AdSelCom; //FTT               //mX
22750: 960 unit unit uPSI_dwsXPlatform;            //DWS
22751: 961 uPSI_AdSocket;                          //mX Turbo Power
22752: 962 uPSI_AdPacket;                          //mX
22753: 963 uPSI_AdPort;                            //mX
22754: 964 uPSI_PathFunc;                          //Inno
22755: 965 uPSI_CmnFunc;                           //Inno
22756: 966 uPSI_CmnFunc2; //Inno Setup             //Inno
22757: 967 unit uPSI_BitmapImage;                  //mX4
22758: 968 unit uPSI_ImageGrabber;                 //mX4
22759: 969 uPSI_SecurityFunc,                      //Inno
22760: 970 uPSI_RedirFunc,                         //Inno
22761: 971 uPSI_FIFO, (MemoryStream)               //mX4
22762: 972 uPSI_Int64Em,                           //Inno
22763: 973 unit uPSI_InstFunc;                     //Inno
22764: 974 unit uPSI_LibFusion;                    //Inno
22765: 975 uPSI_SimpleExpression;                  //Inno
22766: 976 uPSI_unitResourceDetails,               //XN
22767: 977 uPSI_unitResFile,                       //XN
22768: 978 unit uPSI_simpleComport;                //mX4
22769: 979 unit uPSI_AfViewershelpers;             //Async
22770: 980 unit uPSI_Console;                      //mX4
22771: 981 unit uPSI_AnalogMeter;                  //TB
22772: 982 unit uPSI_XPrinter,                     //TB
22773: 983 unit uPSI_IniFiles;                     //VCL
22774: 984 unit uPSI_lazIniFiles;                  //FP
22775: 985 uPSI_testutils;                         //FP
22776: 986 uPSI_ToolsUnit; (DBTests)               //FP
22777: 987 uPSI_fpcunit                            //FP
22778: 988 uPSI_testdecorator;                     //FP
22779: 989 unit uPSI_fpcunittests;                 //FP
22780: 990 unit uPSI_cTCPBuffer;                   //Fundamentals 4
22781: 991 unit uPSI_Glut,                         //Open GL
22782: 992 uPSI_LEDBitmaps,                        //mX4
22783: 993 uPSI_FileClass,                         //Inno
22784: 994 uPSI_FileUtilsClass,                    //mX4
22785: 995 uPSI_ComPortInterface; //Kit           //mX4
22786: 996 unit uPSI_SwitchLed;                    //mX4
22787: 997 unit uPSI_cyDmmCanvas;                  //cY
22788: 998 uPSI_uColorFunctions;                   //DFF
22789: 999 uPSI_uSettings;                         //DFF
22790: 1000 uPSI_cyDebug.pas                       //cY
22791: 1001 uPSI_cyColorMatrix;                     //cY
22792: 1002 unit uPSI_cyCopyFiles;                 //cY
22793: 1003 unit uPSI_cySearchFiles;               //cY
22794: 1004 unit uPSI_cyBaseMeasure;               //cY
22795: 1005 unit uPSI_PJIStreams;                  //DD
22796: 1006 unit uPSI_cyRunTimeResize;             //cY
22797: 1007 unit uPSI_jcontrolutils;               //cY
22798: 1008 unit uPSI_kcMapViewer; (+GEONames)     //kc
22799: 1009 unit uPSI_kcMapViewerDESynapse;        //kc
22800: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
22801: 1011 unit uPSI_LedNumber;                   //TurboPower
22802: 1012 unit uPSI_StStrL;                      //SysTools
22803: 1013 unit uPSI_indGnouMeter;                //LAZ
22804: 1014 unit uPSI_Sensors;                     //LAZ
22805: 1015 unit uPSI_pwmain;  //cgi of powtils    //Pow
22806: 1016 unit uPSI_HTMLUtil;                    //Pow
22807: 1017 unit uPSI_synwrap1; //httpsend         //Pow
22808: 1018 unit StreamWrap1                       //Pow
22809: 1019 unit uPSI_pwmain;                      //Pow
22810: 1020 unit pwtypes                           //Pow
22811: 1021 uPSI_W32VersionInfo                    //LAZ
22812: 1022 unit uPSI_IpAnim;                      //LAZ
22813: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
22814: 1024 unit uPSI_LrtPoTools;                  //LAZ
22815: 1025 unit uPSI_Laz_DOM;                     //LAZ
22816: 1026 unit uPSI_hhAvComp;                    //LAZ
22817: 1027 unit uPSI_GPS2;                        //mX4
22818: 1028 unit uPSI_GPS;                         //mX4
22819: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo//mX4
22820: 1030 unit uPSI_NMEA;     // GPS              //mX4
```

```
22821: 1031 unit uPSI_ScreenThreeDLab;               //mX4
22822: 1032 unit uPSI_Spin;                          //VCL
22823: 1033 unit uPSI_DynaZip;                       //mX4
22824: 1034 unit uPSI_clockExpert;                   //TB
22825: 1035 unit debugLn                            //mX4
22826: 1036 uPSI_SortUtils;                         //Jcl
22827: 1037 uPSI_BitmapConversion;                  //Jcl
22828: 1038 unit uPSI_JclTD32;                      //Jcl
22829: 1039 unit uPSI_ZDbcUtils;                    //Zeos
22830: 1040 unit uPSI_ZScriptParser;               //Zeos
22831: 1041 uPSI_JvIni,                            //JCL
22832: 1042 uPSI_JvFtpGrabber;                     //JCL
22833: 1043 unit uPSI_NeuralNetwork;               //OCL
22834: 1044 unit uPSI_StExpr;                      //SysTools
22835: 1045 unit uPSI_GR32_Geometry;               //GR32
22836: 1046 unit uPSI_GR32_Containers;             //GR32
22837: 1047 unit uPSI_GR32_Backends_VCL,           //GR32
22838: 1048 unit uPSI_StSaturn;//Venus+Mercury+Mars++ //SysTools
22839: 1049 unit uPSI_JclParseUses;                //JCL
22840: 1050 unit uPSI_JvFinalize;                  //JCL
22841: 1051 unit uPSI_panUnit1;                    //GLScene
22842: 1052 unit uPSI_DD83u1;  //Arduino Tester    //mX4
22843:
22844:
22845: ////////////////////////////////////////////////////////////////////////////
22846: //Form Template Library FTL
22847: ////////////////////////////////////////////////////////////////////////////
22848:
22849: 36 FTL For Form Building out of the Script, eg. 399_form_templates.txt
22850:
22851: 045 unit uPSI_VListView                       TFormListView;
22852: 263 unit uPSI_JvProfiler32;                   TProfReport
22853: 270 unit uPSI_ImgList;                        TCustomImageList
22854: 278 unit uPSI_JvImageWindow;                  TJvImageWindow
22855: 317 unit uPSI_JvParserForm;                   TJvHTMLParserForm
22856: 497 unit uPSI_DebugBox;                       TDebugBox
22857: 513 unit uPSI_ImageWin;                       TImageForm, TImageForm2
22858: 514 unit uPSI_CustomDrawTreeView;            TCustomDrawForm
22859: 515 unit uPSI_GraphWin;                      TGraphWinForm
22860: 516 unit uPSI_actionMain;                    TActionForm
22861: 518 unit uPSI_CtlPanel;                      TAppletApplication
22862: 529 unit uPSI_MDIEdit;                       TEditForm
22863: 535 unit uPSI_CoolMain; {browser}           TWebMainForm
22864: 538 unit uPSI_frmExportMain;                TSynexportForm
22865: 585 unit uPSI_usniffer; {//PortScanForm}    TSniffForm
22866: 600 unit uPSI_ThreadForm;                   TThreadSortForm;
22867: 618 unit uPSI_delphi_arduino_Unit1;         TLEDForm
22868: 620 unit uPSI_fplotMain;                    TfplotForm1
22869: 660 unit uPSI_JvDBQueryParamsForm;          TJvQueryParamsDialog
22870: 677 unit uPSI_OptionsFrm;                   TfrmOptions;
22871: 718 unit uPSI_MonForm;                      TMonitorForm
22872: 742 unit uPSI_SimplePortMain;               TPortForm1
22873: 770 unit uPSI_ovccalc;                      TOvcCalculator  //widget
22874: 810 unit uPSI_DbExcept;                     TDbEngineErrorDlg
22875: 812 unit uPSI_GL_actorUnit1;               TglActorForm1  //OpenGL Robot
22876: 846 unit uPSI_synhttp_daemon;               TTCPHttpDaemon, TTCPHttpThrd, TPingThread
22877: 867 unit uPSI_Debug;                        TfrmDebug
22878: 904 unit uPSI_U_HexView;                    THexForm2
22879: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)  TOscfrmMain
22880: 959 unit uPSI_AdSelCom;                     TComSelectForm
22881: 1029 unit uPSI_GPSUDemo;                    TFDemo
22882: 1031 unit uPSI_ScreenThreeDLab;             TFormLab3D
22883: 1051 unit uPSI_panUnit1;                    TPanForm1 //GLScene
22884: 1052 unit uPSI_DD83u1;  {Arduino Tester Frm}  TDD83f1
22885:
22886: ex.:with TEditForm.create(self) do begin
22887:       caption:= 'Template Form Tester';
22888:       FormStyle:= fsStayOnTop;
22889:       with editor do begin
22890:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
22891:         SelStart:= 0;
22892:         Modified:= False;
22893:       end;
22894:     end;
22895:   with TWebMainForm.create(self) do begin
22896:     URLs.Text:= 'http://www.kleiner.ch';
22897:     URLsClick(self); Show;
22898:   end;
22899:   with TSynexportForm.create(self) do begin
22900:     Caption:= 'Synexport HTML RTF tester';
22901:     Show;
22902:   end;
22903:   with TThreadSortForm.create(self) do begin
22904:      showmodal; free;
22905:   end;
22906:   with TCustomDrawForm.create(self) do begin
22907:        width:=820; height:=820;
22908:        image1.height:= 600; //add properties
22909:        image1.picture.bitmap:= image2.picture.bitmap;
```

```
22910:       //SelectionBackground1Click(self) CustomDraw1Click(self);
22911:       Background1.click;
22912:       bitmap1.click; Tile1.click;
22913:       Showmodal;
22914:       Free;
22915:     end;
22916:   with TfplotForm1.Create(self) do begin
22917:     BtnPlotClick(self);
22918:     Showmodal; Free;
22919:   end;
22920:  with TOvcCalculator.create(self) do begin
22921:    parent:= aForm;
22922:   //free;
22923:    setbounds(550,510,200,150);
22924:    displaystr:= 'maXcalc';
22925:  end;
22926:  with THexForm2.Create(self) do begin
22927:     ShowModal;
22928:     Free;
22929:  end;
22930:
22931: function CheckBox: string;
22932: var idHTTP: TIDHTTP;
22933:  begin
22934:    result:= 'version not found';
22935:    if IsInternet then begin
22936:      idHTTP:= TIdHTTP.Create(NIL);
22937:      try
22938:        result:= idHTTP.Get(MXVERSIONFILE2);
22939:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
22940:        if result = MBVER2 then begin
22941:          //output.Font.Style:= [fsbold];
22942:          //Speak(' A new Version '+vstr+' of max box is available! ');
22943:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
22944:        end;  //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
22945:      finally
22946:        idHTTP.Free
22947:    end;
22948:  end;
22949: end;
22950:
22951: //Runtimer Functions Edition
22952: function ApWinExecAndWait32(FileName:PChar; CommandLine:PChar; Visibility:Integer):Integer;
22953: function KillTask(ExeFileName: string): Integer;
22954: procedure KillProcess(hWindowHandle: HWND);
22955: function FindWindowByTitle(WindowTitle: string): Hwnd;
22956: function IntToFloat(i: Integer): double;
22957: function AddThousandSeparator(S: string; myChr: Char): string;
22958: function mciSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
22959: procedure FormAnimation(Sender: TObject; adelay: integer);
22960: procedure LoadResourceFile2(aFile:string; ms:TMemoryStream);
22961: function putBinResTo(binresname: pchar; newpath: string): boolean;
22962: procedure ExecuteHyperlink(Sender: TObject; HyperLinkClick: TJvHyperLinkClickEvent; const LinkName:
      string);');
22963: function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string; MouseX,MouseY:Integer;var
      HyperLink:string):Bool
22964: Function GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD) : DWORD;');
22965: Function GetWindowTask( hWnd : HWND) : THandle');
22966: Function LoadBitmap( hInstance : HINST; lpBitmapName : PChar) : HBITMAP');
22967: Function GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL');
22968: function WinExecAndWait32(FileName: string; Visibility: Integer): Longword;
22969:
22970:   command1:= 'play "'+songpath+'maxbox.wav"';
22971:   command2:= 'play "'+songpath+'moon.wav"';
22972:   SendMCICommand('open waveaudio shareable');   //parallels
22973:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav"');
22974:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav"');
22975:   SendMCICommand('close waveaudio');
22976:
22977: ////////////////////////////////////////////////////////////////////////////
22978: All maXbox Tutorials Table of Content 2014
22979: ////////////////////////////////////////////////////////////////////////////
22980:     Tutorial 00 Function-Coding  (Blix the Programmer)
22981:     Tutorial 01 Procedural-Coding
22982:     Tutorial 02 OO-Programming
22983:     Tutorial 03 Modular Coding
22984:     Tutorial 04 UML Use Case Coding
22985:     Tutorial 05 Internet Coding
22986:     Tutorial 06 Network Coding
22987:     Tutorial 07 Game Graphics Coding
22988:     Tutorial 08 Operating System Coding
22989:     Tutorial 09 Database Coding
22990:     Tutorial 10 Statistic Coding
22991:     Tutorial 11 Forms Coding
22992:     Tutorial 12 SQL DB Coding
22993:     Tutorial 13 Crypto Coding
22994:     Tutorial 14 Parallel Coding
22995:     Tutorial 15 Serial RS232 Coding
22996:     Tutorial 16 Event Driven Coding
```

```
22997:     Tutorial 17 Web Server Coding
22998:     Tutorial 18 Arduino System Coding
22999:     Tutorial 18_3 RGB LED System Coding
23000:     Tutorial 19 WinCOM /Arduino Coding
23001:     Tutorial 20 Regular Expressions RegEx
23002:     Tutorial 21 Android Coding (coming 2013)
23003:     Tutorial 22 Services Programming
23004:     Tutorial 23 Real Time Systems
23005:     Tutorial 24 Clean Code
23006:     Tutorial 25 maXbox Configuration I+II
23007:     Tutorial 26 Socket Programming with TCP
23008:     Tutorial 27 XML & TreeView
23009:     Tutorial 28 DLL Coding (available)
23010:     Tutorial 29 UML Scripting (2014)
23011:     Tutorial 30 Web of Things (2014)
23012:     Tutorial 31 Closures (2014)
23013:     Tutorial 32 SQL Firebird (coming 2014)
23014:     Tutorial 33 Oscilloscope (coming 2015)
23015:     Tutorial 34 GPS Navigation (2014)
23016:     Tutorial 35 Web Box (available)
23017:     Tutorial 36 Unit Testing (coming 2015)
23018:     Tutorial 37 API Coding (coming 2015)
23019:     Tutorial 38 3D Coding (coming 2015)
23020:     Tutorial 39 GEO Map Coding (available)
23021:     Tutorial 39_1 GEO Map Layers Coding (available)
23022:     Tutorial 40 REST Coding (coming 2015)
23023:
23024:
23025: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
23026: using Docu for this file is maxbox_functions_all.pdf
23027: PEP - Pascal Education Program  Low Lib Lab ShellHell
23028:
23029:
23030: http://stackoverflow.com/tags/pascalscript/hot
23031: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
23032: http://sourceforge.net/projects/maXbox   #locs:51620
23033: http://sourceforge.net/apps/mediawiki/maXbox
23034: http://www.blaisepascal.eu/
23035: https://github.com/maxkleiner/maXbox3.git
23036: http://www.heise.de/download/maxbox-1176464.html
23037: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
23038: https://www.facebook.com/pages/Programming-maXbox/166844836691703
23039: http://www.softwareschule.ch/arduino_training.pdf
23040: http://www.delphiarea.com
23041: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
23042: http://entwickler-konferenz.de/2014/speakers/max-kleiner
23043: http://www.heise.de/download/maxbox-1176464.html
23044:
23045:
23046:
23047: All maXbox Examples List
23048: https://github.com/maxkleiner/maXbox3/releases
23049: ********************************************************************************************
23050: 000_pas_baseconvert.txt                          282_fadengraphik.txt
23051: 000_pas_baseconvert.txt_encrypt                  283_SQL_API_messagetimeout.txt
23052: 000_pas_baseconvert.txt_decrypt                  284_SysTools4.txt
23053: 001_1_pas_functest - Kopie.txt                   285_MineForm_GR32.TXT
23054: 001_1_pas_functest.txt                          285_MineForm_GR32main.TXT
23055: 001_1_pas_functest2.txt                         285_MineForm_GR32mainsolution.TXT
23056: 001_1_pas_functest_clx2.txt                     285_MineForm_propas.TXT
23057: 001_1_pas_functest_clx2_2.txt                   285_MineForm_propas2.TXT
23058: 001_1_pas_functest_openarray.txt                285_minesweeper2.TXT
23059: 001_pas_lottogen.txt                            285_Patterns_process.txt
23060: 001_pas_lottogen_template.txt                   286_colormixer_jpeg_charcounter.txt
23061: 001_pas_lottogen.txtcopy                        286_colormixer_jpeg_charcounter2.txt
23062: 002_pas_russianroulette.txt                     287_eventhandling.txt
23063: 002_pas_russianroulette.txtcopy                 287_eventhandling2.txt
23064: 002_pas_russianroulette.txtcopy_decrypt         287_eventhandling2_negpower.txt
23065: 002_pas_russianroulette.txtcopy_encrypt         288_bitblt.txt
23066: 003_pas_motion.txt                              288_bitblt_resize.txt
23067: 003_pas_motion.txtcopy                          289_regression.txt
23068: 004_pas_search.txt                              289_regression2.txt
23069: 004_pas_search_replace.txt                      290_bestofbox.txt
23070: 004_search_replace_allfunctionlist.txt          290_bestofbox2.txt
23071: 005_pas_oodesign.txt                            290_bestofbox3.txt
23072: 005_pas_shelllink.txt                           291_3sort_visual_thread.txt
23073: 006_pas_oobatch.txt                             292_refactoring2.txt
23074: 007_pas_streamcopy.txt                          293_bold_utils.txt
23075: 008_EINMALEINS_FUNC.TXT                         293_ib_utils.txt
23076: 008_explanation.txt                             293_ib_utils_timetest.txt
23077: 008_pas_verwechselt.txt                         294_maxcalc_demo.txt
23078: 008_pas_verwechselt_ibz_bern_func.txt           294_maxcalc_demo2.txt
23079: 008_stack_ibz.TXT                               295_easter_calendar.txt
23080: 009_pas_umlrunner.txt                           295_easter_calendar2.txt
23081: 009_pas_umlrunner_all.txt                       295_easter_combobox.txt
23082: 009_pas_umlrunner_componenttest.txt             297_atomimage.txt
23083: 009_pas_umlrunner_solution.txt                  297_atomimage2.txt
23084: 009_pas_umlrunner_solution_2step.txt            297_atomimage3.txt
23085: 010_pas_oodesign_solution.txt                   297_atomimage4.txt
```

```
23086: 011_pas_puzzlepas_defect.txt
23087: 012_pas_umlrunner_solution.txt
23088: 012_pas_umlrunner_solution2.txt
23089: 013_pas_linenumber.txt
23090: 014_pas_primetest.txt
23091: 014_pas_primetest_first.txt
23092: 014_pas_primetest_sync.txt
23093: 015_pas_designbycontract.txt
23094: 015_pas_designbycontract_solution.txt
23095: 016_pas_searchrec.txt
23096: 017_chartgen.txt
23097: 018_data_simulator.txt
23098: 019_dez_to_bin.txt
23099: 019_dez_to_bin_grenzwert_ibz.txt
23100: 020_proc_feedback.txt
23101: 021_pas_symkey.txt
23102: 021_pas_symkey_solution.txt
23103: 022_pas_filestreams.txt
23104: 023_pas_find_searchrec.txt
23105: 023_pas_pathfind.txt
23106: 024_pas_TFileStream_records.txt
23107: 025_prime_direct.txt
23108: 026_pas_memorystream.txt
23109: 027_pas_shellexecute_beta.txt
23110: 027_pas_shellexecute_solution.txt
23111: 028_pas_dataset.txt
23112: 029_pas_assignfile.txt
23113: 029_pas_assignfile_dragndropexe.txt
23114: 030_palindrome_2.txt
23115: 030_palindrome_tester.txt
23116: 030_pas_recursion.txt
23117: 030_pas_recursion2.txt
23118: 031_pas_hashcode.txt
23119: 032_pas_crc_const.txt
23120: 033_pas_cipher.txt
23121: 033_pas_cipher_def.txt
23122: 033_pas_cipher_file_2_solution.txt
23123: 034_pas_soundbox.txt
23124: 035_pas_crcscript.txt
23125: 035_pas_CRCscript_modbus.txt
23126: 036_pas_includetest.txt
23127: 036_pas_includetest_basta.txt
23128: 037_pas_define_demo32.txt
23129: 038_pas_box_demonstrator.txt
23130: 039_pas_dllcall.txt
23131: 040_paspointer.txt
23132: 040_paspointer_old.txt
23133: 041_pasplotter.txt
23134: 041_pasplotter_plus.txt
23135: 042_pas_kgv_ggt.txt
23136: 043_pas_proceduretype.txt
23137: 044_pas_14queens_solwith14.txt
23138: 044_pas_8queens.txt
23139: 044_pas_8queens_sol2.txt
23140: 044_pas_8queens_solutions.txt
23141: 044_queens_performer.txt
23142: 044_queens_performer2.txt
23143: 044_queens_performer2tester.txt
23144: 045_pas_listhandling.txt
23145: 046_pas_records.txt
23146: 047_pas_modula10.txt
23147: 048_pas_romans.txt
23148: 049_pas_ifdemo.txt
23149: 049_pas_ifdemo_BROKER.txt
23150: 050_pas_primetest2.txt
23151: 050_pas_primetester_thieves.txt
23152: 050_program_starter.txt
23153: 050_program_starter_performance.txt
23154: 051_pas_findtext_solution.txt
23155: 052_pas_text_as_stream.txt
23156: 052_pas_text_as_stream_include.txt
23157: 053_pas_singleton.txt
23158: 054_pas_speakpassword.txt
23159: 054_pas_speakpassword2.txt
23160: 054_pas_speakpassword_searchtest.txt
23161: 055_pas_factorylist.txt
23162: 056_pas_demeter.txt
23163: 057_pas_dirfinder.txt
23164: 058_pas_filefinder.txt
23165: 058_pas_filefinder_pdf.txt
23166: 058_pas_filefinder_screview.txt
23167: 058_pas_filefinder_screview2.txt
23168: 058_pas_filefinder_screview3.txt
23169: 059_pas_timertest.txt
23170: 059_pas_timertest_2.txt
23171: 059_pas_timertest_time_solution.txt
23172: 059_timerobject_starter2.txt
23173: 059_timerobject_starter2_ibz2_async.txt
23174: 059_timerobject_starter2_uml.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT
315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
```

```
23175: 059_timerobject_starter2_uml_main.txt
23176: 059_timerobject_starter4_ibz.txt
23177: 060_pas_datefind.txt
23178: 060_pas_datefind_exceptions2.txt
23179: 060_pas_datefind_exceptions_CHECKTEST.txt
23180: 060_pas_datefind_fulltext.txt
23181: 060_pas_datefind_plus.txt
23182: 060_pas_datefind_plus_mydate.txt
23183: 061_pas_randomwalk.txt
23184: 061_pas_randomwalk_plus.txt
23185: 062_paskorrelation.txt
23186: 063_pas_calculateform.txt
23187: 063_pas_calculateform_2list.txt
23188: 064_pas_timetest.txt
23189: 065_pas_bitcounter.txt
23190: 066_pas_eliza.txt
23191: 066_pas_eliza_include_sol.txt
23192: 067_pas_morse.txt
23193: 068_pas_piezo_sound.txt
23194: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
23195: 069_my_LEDBOX.TXT
23196: 069_pas_ledmatrix.txt
23197: 069_pas_LEDMATRIX_Alphabet.txt
23198: 069_pas_LEDMATRIX_Alphabet_run.txt
23199: 069_pas_LEDMATRIX_Alphabet_tester.txt
23200: 069_PAS_LEDMATRIX_COLOR.TXT
23201: 069_pas_ledmatrix_fixedit.txt
23202: 069_pas_LEDMATRIX_soundbox.txt
23203: 069_pas_LEDMATRIX_soundbox2.txt
23204: 069_Richter_MATRIX.TXT
23205: 070_pas_functionplot.txt
23206: 070_pas_functionplotter2.txt
23207: 070_pas_functionplotter2_mx4.txt
23208: 070_pas_functionplotter2_tester.txt
23209: 070_pas_functionplotter3.txt
23210: 070_pas_functionplotter4.txt
23211: 070_pas_functionplotter_digital.txt
23212: 070_pas_functionplotter_elliptic.txt
23213: 070_pas_function_helmholtz.txt
23214: 070_pas_textcheck_experimental.txt
23215: 071_pas_graphics.txt
23216: 071_pas_graphics_drawsym.txt
23217: 071_pas_graphics_drawsym_save.txt
23218: 071_pas_graphics_random.txt
23219: 072_pas_fractals.txt
23220: 072_pas_fractals_2.txt
23221: 072_pas_fractals_blackhole.txt
23222: 072_pas_fractals_perfomance.txt
23223: 072_pas_fractals_perfomance_new.txt
23224: 072_pas_fractals_perfomance_sharp.txt
23225: 072_pas_fractals_performance.txt
23226: 072_pas_fractals_performance_mX4.txt
23227: 073_pas_forms.txt
23228: 074_pas_chartgenerator.txt
23229: 074_pas_chartgenerator_solution.txt
23230: 074_pas_chartgenerator_solution_back.txt
23231: 074_pas_charts.txt
23232: 075_bitmap_Artwork2.txt
23233: 075_pas_bitmappuzzle.txt
23234: 075_pas_bitmappuzzle24.prod.txt
23235: 075_pas_bitmappuzzle2_prod.txt
23236: 075_pas_bitmappuzzle3.txt
23237: 075_pas_bitmapsolve.txt
23238: 075_pas_bitmap_Artwork.txt
23239: 075_pas_puzzlepas_solution.txt
23240: 076_pas_3dcube.txt
23241: 076_pas_circle.txt
23242: 077_pas_mmshow.txt
23243: 078_pas_pi.txt
23244: 079_pas_3dcube_animation.txt
23245: 079_pas_3dcube_animation4.txt
23246: 079_pas_3dcube_plus.txt
23247: 080_pas_hanoi.txt
23248: 080_pas_hanoi2.txt
23249: 080_pas_hanoi2_file.txt
23250: 080_pas_hanoi2_sol.txt
23251: 080_pas_hanoi2_tester.txt
23252: 080_pas_hanoi2_tester_fast.txt
23253: 080_pas_hanoi3.txt
23254: 081_pas_chartist2.txt
23255: 082_pas_biorythmus.txt
23256: 082_pas_biorythmus_solution.txt
23257: 082_pas_biorythmus_solution_3.txt
23258: 082_pas_biorythmus_test.txt
23259: 083_pas_GITARRE.txt
23260: 083_pas_soundbox_tones.txt
23261: 084_pas_waves.txt
23262: 085_mxsinus_logo.txt
23263: 085_sinus_plot_waves.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docutype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_biginteger.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT
362_maxonmotor2.TXT
363_compress_services.txt
363_compress_services2.txt
364_pdf_services.txt
365_memorystream.txt
365_memorystream2.txt
365_memorystream_test.txt
365_U_HexView.txt
366_mp3player.txt
366_mp3player2.txt
366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maXbook_v4tester.txt
372stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
```

23264: 086_pas_graph_arrow_heart.txt
23265: 087_bitmap_loader.txt
23266: 087_pas_bitmap_solution.txt
23267: 087_pas_bitmap_solution2.txt
23268: 087_pas_bitmap_subimage.txt
23269: 087_pas_bitmap_test.txt
23270: 088_pas_soundbox2_mp3.txt
23271: 088_pas_soundbox_mp3.txt
23272: 088_pas_sphere_2.txt
23273: 089_pas_gradient.txt
23274: 089_pas_maxland2.txt
23275: 090_pas_sudoku4.txt
23276: 090_pas_sudoku4_2.txt
23277: 091_pas_cube4.txt
23278: 092_pas_statistics4.txt
23279: 093_variance.txt
23280: 093_variance_debug.txt
23281: 094_pas_daysold.txt
23282: 094_pas_stat_date.txt
23283: 095_pas_ki_simulation.txt
23284: 096_pas_geisen_problem.txt
23285: 096_pas_montyhall_problem.txt
23286: 097_lotto_proofofconcept.txt
23287: 097_pas_lottocombinations_beat_plus.txt
23288: 097_pas_lottocombinations_beat_plus2.txt
23289: 097_pas_lottocombinations_universal.txt
23290: 097_pas_lottosimulation.txt
23291: 098_pas_chartgenerator_plus.txt
23292: 099_pas_3D_show.txt
23293: 200_big_numbers.txt
23294: 200_big_numbers2.txt
23295: 201_streamload_xml.txt
23296: 202_systemcheck.txt
23297: 203_webservice_simple_intftester.txt
23298: 204_webservice_simple.txt
23299: 205_future_value_service.txt
23300: 206_DTD_string_functions.txt
23301: 207_ibz2_async_process.txt
23302: 208_crc32_hash.txt
23303: 209_cryptohash.txt
23304: 210_public_private.txt
23305: 210_public_private_cryptosystem.txt
23306: 211_wipe_pattern.txt
23307: 211_wipe_pattern2.txt
23308: 211_wipe_pattern_solution.txt
23309: 212_pas_statisticmodule4.TXT
23310: 212_pas_statisticmoduletxt.TXT
23311: 212_statisticmodule4.txt
23312: 213_pas_BBP_Algo.txt
23313: 214_mxdocudemo.txt
23314: 214_mxdocudemo2.txt
23315: 214_mxdocudemo3.txt
23316: 215_hints_test.TXT
23317: 216_warnings_test.TXT
23318: 217_pas_heartbeat.txt
23319: 218_biorythmus_studio.txt
23320: 219_cipherbox.txt
23321: 219_crypt_source_comtest_solution.TXT
23322: 220_cipherbox_form.txt
23323: 220_cipherbox_form2.txt
23324: 221_bcd_explain.txt
23325: 222_memoform.txt
23326: 223_directorybox.txt
23327: 224_dialogs.txt
23328: 225_sprite_animation.txt
23329: 226_ASCII_Grid2.TXT
23330: 227_animation.txt
23331: 227_animation2.txt
23332: 228_android_calendar.txt
23333: 229_android_game.txt
23334: 229_android_game_tester.txt
23335: 230_DataProvider.txt
23336: 230_DataSetProvider.txt
23337: 230_DataSetXMLBackupScholz.txt
23338: 231_DBGrid_access.txt
23339: 231_DBGrid_XMLaccess.txt
23340: 231_DBGrid_XMLaccess2.txt
23341: 231_DBGrid_XMLaccess_locatetester.txt
23342: 231_DBGrid_XML_CDS_local.txt
23343: 232_outline.txt
23344: 232_outline_2.txt
23345: 233_modular_form.txt
23346: 234_debugoutform.txt
23347: 235_fastform.TXT
23348: 236_componentpower.txt
23349: 236_componentpower_back.txt
23350: 237_pas_4forms.txt
23351: 238_lottogen_form.txt
23352: 239_pas_sierpinski.txt

377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefunctesterfilemon.txt
380_coolfunc.txt
380_coolfunc2.txt
380_coolfunc_tester.txt
381_bitcoin_simulation.txt
382_GRMath.TXT
382_GRMath_PI_Proof.TXT
382_GRMath_Riemann.TXT
383_MDAC_DCOM.txt
384_TeamViewerID.TXT
386_InternetRadio.TXT
387_fulltextfinder.txt
387_fulltextfinder_cleancode.txt
387_fulltextfinder_fast.txt
387_fulltext_getscripttest.txt
388_TCPServerSock.TXT
388_TCPServerSock2.TXT
388_TCPServerSockClient.TXT
389_TAR_Archive.TXT
389_TAR_Archive_test.TXT
389_TAR_Archive_test2.TXT
390_Callback3.TXT
390_Callback3Rec.TXT
390_CallbackClean.TXT
390_StringlistHTML.TXT
391_ToDo_List.TXT
392_Barcode.TXT
392_Barcode2.TXT
392_Barcode23.TXT
392_Barcode2scholz.TXT
392_Barcode3scholz.TXT
393_QRCode.TXT
393_QRCode2.TXT
393_QRCode2Direct.TXT
393_QRCode2DirectIndy.TXT
393_QRCode2Direct_detlef.TXT
393_QRCode3.TXT
394_networkgraph.TXT
394_networkgraph_depwalkutilstest.TXT
394_networkgraph_depwalkutilstest2.TXT
395_USBController.TXT
396_Sort.TXT
397_Hotlog.TXT
397_Hotlog2.TXT
398_ustrings.txt
399_form_templates.txt
400_fplottchart.TXT
400_fplottchart2.TXT
400_fplottchart2teetest.TXT
400_QRCodeMarket.TXT
401_tfilerun.txt
402_richedit2.txt
403_outlookspy.txt
404_simplebrowser.txt
405_datefinder_today.txt
406_portscan.txt
407_indydemo.txt
408_testroboter.txt
409_excel_control.txt
410_keyboardevent.txt
411_json_test.txt
412_Zeosutils.txt
413_listview2.txt
414_avrdude_flash.txt
415_avrdude_writehex.txt
416_sonar_startscriptEKON.TXT
416_sonar_startscriptEKON_reporting.TXT
417_GRMath_PI_Proof2.TXT
418_functional_paradigm.txt
419_archimedes_spiral.txt
419_archimedes_spiral2.txt
420_archimedes_arduino.txt
420_Lissajous.txt
421_PI_Power.TXT
421_PI_Power2.TXT
422_world_bitboxx.txt
423_game_of_life.TXT
423_game_of_life2.TXT
423_game_of_life3.TXT
423_game_of_life3_test.TXT
423_game_of_life4.TXT
423_game_of_life4_kryptum.TXT
424_opengl_tester.txt

23353: 239_pas_sierpinski2.txt
23354: 240_unitGlobal_tester.txt
23355: 241_db3_sql_tutorial.txt
23356: 241_db3_sql_tutorial2.txt
23357: 241_db3_sql_tutorial2fix.txt
23358: 241_db3_sql_tutorial3.txt
23359: 241_db3_sql_tutorial3connect.txt
23360: 241_db3_sql_tutorial3_fptest.txt
23361: 241_RTL_SET2.txt
23362: 241_RTL_SET2_tester.txt
23363: 242_Component_Control.txt
23364: 243_tutorial_loader.txt
23365: 244_script_loader_loop.txt
23366: 245_formapp2.txt
23367: 245_formapp2_tester.txt
23368: 245_formapp2_testerX.txt
23369: 246_httpapp.txt
23370: 247_datecalendar.txt
23371: 248_ASCII_Grid2_sorted.TXT
23372: 249_picture_grid.TXT
23373: 250_tipsandtricks2.txt
23374: 250_tipsandtricks3.txt
23375: 250_tipsandtricks3api.txt
23376: 250_tipsandtricks3_admin_elevation.txt
23377: 250_tipsandtricks3_tester.txt
23378: 250_tipsandtricks4_tester.txt
23379: 250_tipsandtricks4_tester2.txt
23380: 251_compare_noise_gauss.txt
23381: 251_whitenoise.txt
23382: 251_whitenoise2.txt
23383: 252_hilbert_turtle.txt
23384: 252_pas_hilbert.txt
23385: 253_opearatingsystem3.txt
23386: 254_dynarrays.txt
23387: 255_einstein.txt
23388: 256_findconsts_of_EXE.txt
23389: 256_findfunctions2_of_EXE.txt
23390: 256_findfunctions2_of_EXEaverp.txt
23391: 256_findfunctions2_of_EXEspec.txt
23392: 256_findfunctions3.txt
23393: 256_findfunctions_of_EXE.txt
23394: 257_AES_Cipher.txt
23395: 258_AES_cryptobox.txt
23396: 258_AES_cryptobox2.txt
23397: 258_AES_cryptobox2_passdlg.txt
23398: 259_AES_crypt_directory.txt
23399: 260_sendmessage_2.TXT
23400: 260_sendmessage_beta.TXT
23401: 261_probability.txt
23402: 262_mxoutputdemo4.txt
23403: 263_async_sound.txt
23404: 264_vclutils.txt
23405: 264_VCL_utils2.txt
23406: 265_timer_API.txt
23407: 266_serial_interface.txt
23408: 266_serial_interface2.txt
23409: 266_serial_interface3.txt
23410: 267_ackermann_rec.txt
23411: 267_ackermann_variants.txt
23412: 268_DBGrid_tree.txt
23413: 269_record_grid.TXT
23414: 270_Jedi_FunctionPower.txt
23415: 270_Jedi_FunctionPowertester.txt
23416: 271_closures_study.txt
23417: 271_closures_study_workingset2.txt
23418: 272_pas_function_show.txt
23419: 273_pas_function_show2.txt
23420: 274_library_functions.txt
23421: 275_turtle_language.txt
23422: 275_turtle_language_save.txt
23423: 276_save_algo.txt
23424: 276_save_algo2.txt
23425: 277_functionsfor39.txt
23426: 278_DB_Dialogs.TXT
23427: 279_hexer2.TXT
23428: 279_hexer2macro.TXT
23429: 279_hexer2macroback.TXT
23430: 280_UML_process.txt
23431: 280_UML_process_knabe2.txt
23432: 280_UML_process_knabe3.txt
23433: 280_UML_process_TIM_Botzenhardt.txt
23434: 280_UML_TIM_Seitz.txt
23435: 281_picturepuzzle.txt
23436: 281_picturepuzzle2.txt
23437: 281_picturepuzzle3.txt
23438: 281_picturepuzzle4.txt
23439: 479_inputquery.txt
23440: 480_regex_pathfinder2.txt
23441: 482 processPipe.txt

425_reversi_game.txt
426_IBUtils.TXT
427_IBDatabase.TXT
428_SortGrid.TXT
429_fileclass.txt
430_fileoperation.txt
430_fileoperation_tester.txt
431_performance_index.txt
432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
437_WinAPItop_Firebirdtester.txt
438_OvcInternational.txt
439_AsyncFreeDemo.txt
439_AsyncFreeDemoForm.txt
440_DLL_Tutor.txt
440_DLL_Tutor2.txt
440_XML_Tutor.txt
440_XML_Tutor2.txt
441_make_app.txt
442_arduino_rgb_led.txt
443_webserver_arduino_rgb_light.txt
443_webserver_arduino_rgb_light4.txt
444_webserver_arduino3ibz_rgb_led_basta.txt
445_datagrid.txt
445_datagrid2.txt
445_datagrid_android_arduino.txt
446_arduino_timer.txt
447_patternFrm_mx3.txt
448_Synapse.txt
448_Synapse2.txt
449_dweb_start_tester.txt
450_Synapse_HTTPS.txt
450_Synapse_Mime.txt
450_Synapse_ScanPing.txt
451_ocx_player.txt
451_OCX_WinPlayer2.txt
452_dbtreeview.txt
452_dbtreeview2.txt
453_stdfuncs.txt
454_fileStream.txt
455_functionfun.txt
455_functionfun2.txt
455_functionfun2_test.txt
457_ressource_grid.txt
458_atomimageX.txt
459_cindyfunc.txt
459_cindyfunc2.txt
460_TopTenFunctions.txt
461_sqlform_calwin.txt
462_caesarcipher.txt
463_global_exception.txt
464_function_procedure.txt
464_function_procedure2.txt
464_function_procedure3.txt
465_U_HexView.txt
466_moon.txt
466_moon_inputquery.txt
4671_cardmagic.txt
467_helmholtz_graphic.txt
468_URLMon.txt
468_URLMon2.txt
469_formarrow.txt
469_formarrow_datepicker.txt
469_formarrow_datepicker_ibz_result.txt
469_ibzresult.txt
470_DFFUtils_compiled.txt
470_DFFUtils_ScrollingLED.txt
470_Oscilloscope.txt
470_Oscilloscope_code.txt
471_cardmagic.txt
471_cardmagic2.txt
472_allcards.TXT
473_comboset.txt
474_wakeonlan.txt
474_wakeonlan2.txt
476_getscripttest.txt
477_filenameonly.txt
480_regex_pathfinder.txt
481_processList.txt
482 processPipeGCC.txt

```
23442: 483_PathFuncTest_mX.txt                    484_filefinder3.txt
23443: 485_InnoFunc.txt                           486_VideoGrabber.txt
23444: 487_asyncKeyState.txt                      488_asyncTerminal.txt
23445: 489_simpleComport.txt                      490_webCamproc.txt
23446: 491_analogmeter.txt                        492_snowflake2.txt
23447: 493_gadgets.txt                            495_fourierfreq.txt
23448: 496_InstallX.txt                           497_LED.txt
23449: 498_UnitTesting.txt                        499_mulu42.txt
23450: 500_diceoflifes.txt                        501_firebird_datasnap_tests.txt
23451: 502_findalldocs.txt                        503_led_switch.txt
23452: 504_fileclass.txt                          505_debug.txt
23453: 506_colormatrix.txt                        507_derutils.txt
23454: 508_simplecomportmorse.txt                 509_GEOMap2.txt
23455: 509_509_GEOMap2_SReverse.TXT               510_510_bonn_gpsdata_mX4.pas
23456: 511_LEDLabel.txt                          512_LED_moon.txt
23457: 513_StreamIntegration.txt                 514_LED_moon2.txt
23458: 515_ledclock3.txt                         516_mapview.txt
23459: 517_animation7.txt                        518_sensors_meter.txt
23460: 519_powtils.txt                           520_run_bytecode.txt
23461: 521_iputils2.txt                          522_getgeocode.txt
23462: 523_NMEA.txt                             524_NAV_Utils.txt
23463: 525_GEO84s.txt                           526_Compass_meter.txt
23464: 527_GPSDemo.txt                          528_linescount.txt
23465: 529_profilertest.txt                      530_3DLab.txt
23466: 531_profilertest.txt                      532_mcicommand.txt
23467: 533_syncasync_demo.txt                    534_arduino_cockpit.TXT
23468: 535_Battleship3.pas                       536_ressource_grid2.txt
23469: 537_iniplus.TXT                           538_shellbatch.txt
23470: 539_timeturtle123.txt                     540_NeuralNetwork.pas
23471: 541_webserver_arduino_motorturtle.txt     542_arduino_sound.txt
23472: 543_MATH_TurboP.PAS                       544_UTIL01.PAS
23473: 545_strips.TXT                           546_fourier3.pas
23474: 547_regexmaster.TXT                      548_STExpressions.TXT
23475: 549_3D_Panorama.txt                      550_Expressions.TXT - 550_ADO_OLEDB.txt
23476: 551_ArduinoTester.txt                    552_WaitExec32
23477:
23478:
23479: Web Script Examples:
23480:
23481: http://www.softwareschule.ch/examples/performer.txt';
23482: http://www.softwareschule.ch/examples/turtle.txt';
23483: http://www.softwareschule.ch/examples/SQLExport.txt';
23484: http://www.softwareschule.ch/examples/Richter.txt';
23485: http://www.softwareschule.ch/examples/checker.txt';
23486: http://www.softwareschule.ch/examples/demoscript.txt';
23487: http://www.softwareschule.ch/examples/ibzresult.txt';
23488: http://www.softwareschule.ch/examples/performindex.txt
23489: http://www.softwareschule.ch/examples/processlist.txt
23490: http://www.softwareschule.ch/examples/game.txt
23491: http://www.softwareschule.ch/examples/GEOGPS.txt
23492: http://www.softwareschule.ch/examples/turtle2.txt
23493: http://www.softwareschule.ch/examples/asyncterminal.txt
23494:
23495:
23496:
23497: Delphi Basics Run Time Library listing
23498: ********************************************************************************
23499: A
23500: Compiler Directive $A Determines whether data is aligned or packed
23501: Compiler Directive $Align Determines whether data is aligned or packed
23502: Compiler Directive $AppType Determines the application type : GUI or Console
23503: Procedure SysUtils Abort Aborts the current processing with a silent exception
23504: Function System Abs Gives the absolute value of a number (-ve sign is removed)
23505: Directive Abstract Defines a class method only implemented in subclasses
23506: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
23507: Function System Addr Gives the address of a variable, function or procedure
23508: Keyword And Boolean and or bitwise and of two arguments
23509: Type System AnsiChar A character type guaranteed to be 8 bits in size
23510: Function SysUtils AnsiCompareStr Compare two strings for equality
23511: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
23512: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
23513: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
23514: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
23515: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
23516: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
23517: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
23518: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
23519: Function StrUtils AnsiPos Find the position of one string in another
23520: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
23521: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
23522: Function StrUtils AnsiRightStr Extracts characters from the right of a string
23523: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
23524: Type System AnsiString A data type that holds a string of AnsiChars
23525: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
23526: Procedure System Append Open a text file to allow appending of text to the end
23527: Procedure SysUtils AppendStr Concatenate one string onto the end of another
23528: Function Math ArcCos The Arc Cosine of a number, returned in radians
23529: Function Math ArcSin The Arc Sine of a number, returned in radians
23530: Function System ArcTan The Arc Tangent of a number, returned in radians
```

```
23531: Keyword Array A data type holding indexable collections of data
23532: Keyword As Used for casting object references
23533: Procedure System Assign Assigns a file handle to a binary or text file
23534: Function System Assigned Returns true if a reference is not nil
23535: Procedure System AssignFile Assigns a file handle to a binary or text file
23536: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
23537:
23538: B
23539: Compiler Directive $B Whether to short cut and and or operations
23540: Compiler Directive $BoolEval Whether to short cut and and or operations
23541: Procedure SysUtils Beep Make a beep sound
23542: Keyword Begin Keyword that starts a statement block
23543: Function System BeginThread Begins a separate thread of code execution
23544: Procedure System BlockRead Reads a block of data records from an untyped binary file
23545: Procedure System BlockWrite Writes a block of data records to an untyped binary file
23546: Type System Boolean Allows just True and False values
23547: Function Classes Bounds Create a TRect value from top left and size values
23548: Procedure System Break Forces a jump out of a single loop
23549: Type System Byte An integer type supporting values 0 to 255
23550:
23551: C
23552: Type System Cardinal The basic unsigned integer type
23553: Keyword Case A mechanism for acting upon different values of an Ordinal
23554: Function StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
23555: Function SysUtils ChangeFileExt Change the extension part of a file name
23556: Type System Char Variable type holding a single character
23557: Procedure System ChDir Change the working drive plus path for a specified drive
23558: Function System Chr Convert an integer into a character
23559: Keyword Class Starts the declaration of a type of object class
23560: Procedure System Close Closes an open file
23561: Procedure System CloseFile Closes an open file
23562: Variable System CmdLine Holds the execution text used to start the current program
23563: Type System Comp A 64 bit signed integer
23564: Function SysUtils CompareStr Compare two strings to see which is greater than the other
23565: Function SysUtils CompareText Compare two strings for equality, ignoring case
23566: Function Math CompareValue Compare numeric values with a tolerance
23567: Function System Concat Concatenates one or more strings into one string
23568: Keyword Const Starts the definition of fixed data values
23569: Keyword Constructor Defines the method used to create an object from a class
23570: Procedure System Continue Forces a jump to the next iteration of a loop
23571: Function ConvUtils Convert Convert one measurement value to another
23572: Function System Copy Create a copy of part of a string or an array
23573: Function System Cos The Cosine of a number
23574: Function SysUtils CreateDir Create a directory
23575: Type System Currency A floating point type with 4 decimals used for financial values
23576: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
23577: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
23578: Variable SysUtils CurrencyString The currency string used in currency display functions
23579: Function SysUtils CurrToStr Convert a currency value to a string
23580: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
23581:
23582: D
23583: Compiler Directive $D Determines whether application debug information is built
23584: Compiler Directive $DebugInfo Determines whether application debug information is built
23585: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
23586: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
23587: Function SysUtils Date Gives the current date
23588: Variable SysUtils DateSeparator The character used to separate display date fields
23589: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
23590: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
23591: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
23592: Function SysUtils DateToStr Converts a TDateTime date value to a string
23593: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
23594: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
23595: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
23596: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
23597: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
23598: Function DateUtils DaysInAMonth Gives the number of days in a month
23599: Function DateUtils DaysInAYear Gives the number of days in a year
23600: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
23601: Procedure System Dec Decrement an ordinal variable
23602: Variable SysUtils DecimalSeparator The character used to display the decimal point
23603: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
23604: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
23605: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
23606: Directive Default Defines default processing for a property
23607: Function Math DegToRad Convert a degrees value to radians
23608: Procedure System Delete Delete a section of characters from a string
23609: Function SysUtils DeleteFile Delete a file specified by its file name
23610: Keyword Destructor Defines the method used to destroy an object
23611: Function SysUtils DirectoryExists Returns true if the given directory exists
23612: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
23613: Function SysUtils DiskSize Gives the size in bytes of a specified drive
23614: Procedure System Dispose Dispose of storage used by a pointer type variable
23615: Keyword Div Performs integer division, discarding the remainder
23616: Keyword Do Defines the start of some controlled action
23617: Type System Double A floating point type supporting about 15 digits of precision
23618: Keyword DownTo Prefixes an decremental for loop target value
23619: Function StrUtils DupeString Creates a string containing copies of a substring
```

```
23620: Directive Dynamic Allows a class method to be overriden in derived classes
23621:
23622: E
23623: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
23624: Compiler Directive $EndIf Terminates conditional code compilation
23625: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
23626: Keyword Else Starts false section of if, case and try statements
23627: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
23628: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
23629: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
23630: Keyword End Keyword that terminates statement blocks
23631: Function DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
23632: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
23633: Procedure System EndThread Terminates a thread with an exit code
23634: Function System Eof Returns true if a file opened with Reset is at the end
23635: Function System Eoln Returns true if the current text file is pointing at a line end
23636: Procedure System Erase Erase a file
23637: Variable System ErrorAddr Sets the error address when an application terminates
23638: Keyword Except Starts the error trapping clause of a Try statement
23639: Procedure System Exclude Exclude a value in a set variable
23640: Procedure System Exit Exit abruptly from a function or procedure
23641: Variable System ExitCode Sets the return code when an application terminates
23642: Function System Exp Gives the exponent of a number
23643: Directive System Export Makes a function or procedure in a DLL externally available
23644: Type System Extended The floating point type with the highest capacity and precision
23645: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
23646: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
23647: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
23648: Function SysUtils ExtractFileName Extracts the name part of a full file name
23649: Function SysUtils ExtractFilePath Extracts the path part of a full file name
23650:
23651: F
23652: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
23653: Keyword File Defines a typed or untyped file
23654: Function SysUtils FileAge Get the last modified date/time of a file without opening it
23655: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
23656: Function SysUtils FileExists Returns true if the given file exists
23657: Function SysUtils FileGetAttr Gets the attributes of a file
23658: Variable System FileMode Defines how Reset opens a binary file
23659: Function System FilePos Gives the file position in a binary or text file
23660: Function SysUtils FileSearch Search for a file in one or more directories
23661: Function SysUtils FileSetAttr Sets the attributes of a file
23662: Function SysUtils FileSetDate Set the last modified date and time of a file
23663: Function System FileSize Gives the size in records of an open file
23664: Procedure System FillChar Fills out a section of storage with a fill character or byte value
23665: Keyword Finally Starts the unconditional code section of a Try statement
23666: Function SysUtils FindClose Closes a successful FindFirst file search
23667: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
23668: Function SysUtils FindFirst Finds all files matching a file mask and attributes
23669: Function SysUtils FindNext Find the next file after a successful FindFirst
23670: Function SysUtils FloatToStr Convert a floating point value to a string
23671: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
23672: Procedure System Flush Flushes buffered text file data to the file
23673: Keyword For Starts a loop that executes a finite number of times
23674: Function SysUtils ForceDirectories Create a new path of directories
23675: Function SysUtils Format Rich formatting of numbers and text into a string
23676: Function SysUtils FormatCurr Rich formatting of a currency value into a string
23677: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
23678: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
23679: Function System Frac The fractional part of a floating point number
23680: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
23681: Procedure System FreeMem Free memory storage used by a variable
23682: Keyword System Function Defines a subroutine that returns a value
23683:
23684: G
23685: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
23686: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
23687: Function System GetLastError Gives the error code of the last failing Windows API call
23688: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
23689: Function System GetMem Get a specified number of storage bytes
23690: Keyword Goto Forces a jump to a label, regardless of nesting
23691:
23692: H
23693: Compiler Directive $H Treat string types as AnsiString or ShortString
23694: Compiler Directive $Hints Determines whether Delphi shows compilation hints
23695: Procedure System Halt Terminates the program with an optional dialog
23696: Function System Hi Returns the hi-order byte of a (2 byte) Integer
23697: Function System High Returns the highest value of a type or variable
23698:
23699: I
23700: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
23701: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
23702: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
23703: Compiler Directive $IfOpt Tests for the state of a Compiler directive
23704: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
23705: Compiler Directive $IOChecks When on, an IO operation error throws an exception
23706: Keyword If Starts a conditional expression to determine what to do next
23707: Keyword Implementation Starts the implementation (code) section of a Unit
23708: Keyword In Used to test if a value is a member of a set
```

```
23709: Procedure System Inc Increment an ordinal variable
23710: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
23711: Procedure System Include Include a value in a set variable
23712: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
23713: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
23714: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
23715: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
23716: Function DateUtils IncYear Increments a TDateTime variable by a number of years
23717: Directive Index Principally defines indexed class data properties
23718: Constant Math Infinity Floating point value of infinite size
23719: Keyword Inherited Used to call the parent class constructor or destructor method
23720: Variable System Input Defines the standard input text file
23721: Function Dialogs InputBox Display a dialog that asks for user text input, with default
23722: Function Dialogs InputQuery Display a dialog that asks for user text input
23723: Procedure System Insert Insert a string into another string
23724: Function System Int The integer part of a floating point number as a float
23725: Type System Int64 A 64 bit sized integer - the largest in Delphi
23726: Type System Integer The basic Integer type
23727: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
23728: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
23729: Function SysUtils IntToStr Convert an integer into a string
23730: Function System IOResult Holds the return code of the last I/O operation
23731: Keyword Is Tests whether an object is a certain class or ascendant
23732: Function Math IsInfinite Checks whether a floating point number is infinite
23733: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
23734: Function System IsMultiThread Returns true if the code is running multiple threads
23735: Function Math IsNaN Checks to see if a floating point number holds a real number
23736:
23737: L
23738: Compiler Directive $L Determines what application debug information is built
23739: Compiler Directive $LocalSymbols Determines what application debug information is built
23740: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
23741: Function SysUtils LastDelimiter Find the last position of selected characters in a string
23742: Function System Length Return the number of elements in an array or string
23743: Function System Ln Gives the natural logarithm of a number
23744: Function System Lo Returns the low-order byte of a (2 byte) Integer
23745: Function Math Log10 Gives the log to base 10 of a number
23746: Variable SysUtils LongDateFormat Long version of the date to string format
23747: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
23748: Type System LongInt An Integer whose size is guaranteed to be 32 bits
23749: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
23750: Variable SysUtils LongTimeFormat Long version of the time to string format
23751: Type System LongWord A 32 bit unsigned integer
23752: Function System Low Returns the lowest value of a type or variable
23753: Function SysUtils LowerCase Change upper case characters in a string to lower case
23754:
23755: M
23756: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
23757: Function Math Max Gives the maximum of two integer values
23758: Constant System MaxInt The maximum value an Integer can have
23759: Constant System MaxLongInt The maximum value an LongInt can have
23760: Function Math Mean Gives the average for a set of numbers
23761: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
23762: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
23763: Function Math Min Gives the minimum of two integer values
23764: Constant SysUtils MinsPerDay Gives the number of minutes in a day
23765: Procedure System MkDir Make a directory
23766: Keyword Mod Performs integer division, returning the remainder
23767: Constant SysUtils MonthDays Gives the number of days in a month
23768: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
23769: Procedure System Move Copy bytes of data from a source to a destination
23770:
23771: N
23772: Constant Math NaN Not a real number
23773: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
23774: Procedure System New Create a new pointer type variable
23775: Constant System Nil A pointer value that is defined as undetermined
23776: Keyword Not Boolean Not or bitwise not of one arguments
23777: Function SysUtils Now Gives the current date and time
23778: Variable Variants Null A variable that has no value
23779:
23780: O
23781: Compiler Directive $O Determines whether Delphi optimises code when compiling
23782: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
23783: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
23784: Keyword System Object Allows a subroutine data type to refer to an object method
23785: Function System Odd Tests whether an integer has an odd value
23786: Keyword Of Linking keyword used in many places
23787: Keyword On Defines exception handling in a Try Except clause
23788: Keyword Or Boolean or or bitwise or of two arguments
23789: Function System Ord Provides the Ordinal value of an integer, character or enum
23790: Directive Out Identifies a routine parameter for output only
23791: Variable System Output Defines the standard output text file
23792: Directive Overload Allows 2 or more routines to have the same name
23793: Directive Override Defines a method that replaces a virtual parent class method
23794:
23795: P
23796: Keyword Packed Compacts complex data types into minimal storage
23797: Type System PAnsiChar A pointer to an AnsiChar value
```

```
23798: Type System PAnsiString Pointer to an AnsiString value
23799: Function System ParamCount Gives the number of parameters passed to the current program
23800: Function System ParamStr Returns one of the parameters used to run the current program
23801: Type System PChar A pointer to an Char value
23802: Type System PCurrency Pointer to a Currency value
23803: Type System PDateTime Pointer to a TDateTime value
23804: Type System PExtended Pointer to a Extended floating point value
23805: Function System Pi The mathematical constant
23806: Type System PInt64 Pointer to an Int64 value
23807: Function Classes Point Generates a TPoint value from X and Y values
23808: Type System Pointer Defines a general use Pointer to any memory based data
23809: Function Classes PointsEqual Compares two TPoint values for equality
23810: Function System Pos Find the position of one string in another
23811: Function System Pred Decrement an ordinal variable
23812: Function Printers Printer Returns a reference to the global Printer object
23813: Directive Private Starts the section of private data and methods in a class
23814: Keyword System Procedure Defines a subroutine that does not return a value
23815: Procedure FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
23816: Keyword System Program Defines the start of an application
23817: Function Dialogs PromptForFileName Shows a dialog allowing the user to select a file
23818: Keyword System Property Defines controlled access to class fields
23819: Directive Protected Starts a section of class private data accesible to sub-classes
23820: Type System PShortString A pointer to an ShortString value
23821: Type System PString Pointer to a String value
23822: Function Types PtInRect Tests to see if a point lies within a rectangle
23823: Directive Public Starts an externally accessible section of a class
23824: Directive Published Starts a published externally accessible section of a class
23825: Type System PVariant Pointer to a Variant value
23826: Type System PWideChar Pointer to a WideChar
23827: Type System PWideString Pointer to a WideString value
23828:
23829: Q
23830: Compiler Directive $Q Determines whether Delphi checks integer and enum bounds
23831:
23832: R
23833: Compiler Directive $R Determines whether Delphi checks array bounds
23834: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
23835: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
23836: Compiler Directive $Resource Defines a resource file to be included in the application linking
23837: Function Math RadToDeg Converts a radian value to degrees
23838: Keyword Raise Raise an exception
23839: Function System Random Generate a random floating point or integer number
23840: Procedure System Randomize Reposition the Random number generator next value
23841: Function Math RandomRange Generate a random integer number within a supplied range
23842: Variable System RandSeed Reposition the Random number generator next value
23843: Procedure System Read Read data from a binary or text file
23844: Procedure System ReadLn Read a complete line of data from a text file
23845: Type System Real A floating point type supporting about 15 digits of precision
23846: Type System Real48 The floating point type with the highest capacity and precision
23847: Procedure System ReallocMem Reallocate an existing block of storage
23848: Function DateUtils RecodeDate Change only the date part of a TDateTime variable
23849: Function DateUtils RecodeTime Change only the time part of a TDateTime variable
23850: Keyword Record A structured data type - holding fields of data
23851: Function Classes Rect Create a TRect value from 2 points or 4 coordinates
23852: Function SysUtils RemoveDir Remove a directory
23853: Procedure System Rename Rename a file
23854: Function SysUtils RenameFile Rename a file or directory
23855: Keyword Repeat Repeat statements until a ternmination condition is met
23856: Procedure SysUtils ReplaceDate Change only the date part of a TDateTime variable
23857: Procedure SysUtils ReplaceTime Change only the time part of a TDateTime variable
23858: Procedure System Reset Open a text file for reading, or binary file for read/write
23859: Variable System Result A variable used to hold the return value from a function
23860: Procedure System ReWrite Open a text or binary file for write access
23861: Procedure System RmDir Remove a directory
23862: Function System Round Rounds a floating point number to an integer
23863: Procedure System RunError Terminates the program with an error dialog
23864:
23865: S
23866: Constant SysUtils SecsPerDay Gives the number of seconds in a day
23867: Procedure System Seek Move the pointer in a binary file to a new record position
23868: Function System SeekEof Skip to the end of the current line or file
23869: Function System SeekEoln Skip to the end of the current line or file
23870: Function FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
23871: Variable System Self Hidden parameter to a method - refers to the containing object
23872: Keyword Set Defines a set of up to 255 distinct values
23873: Function SysUtils SetCurrentDir Change the current directory
23874: Procedure System SetLength Changes the size of a string, or the size(s) of an array
23875: Procedure System SetString Copies characters from a buffer into a string
23876: Keyword Shl Shift an integer value left by a number of bits
23877: Variable SysUtils ShortDateFormat Compact version of the date to string format
23878: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
23879: Type System ShortInt An integer type supporting values -128 to 127
23880: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
23881: Type System ShortString Defines a string of up to 255 characters
23882: Variable SysUtils ShortTimeFormat Short version of the time to string format
23883: Procedure Dialogs ShowMessage Display a string in a simple dialog with an OK button
23884: Procedure Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
23885: Procedure Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
23886: Keyword Shr Shift an integer value right by a number of bits
```

```
23887: Function System Sin The Sine of a number
23888: Type System Single The smallest capacity and precision floating point type
23889: Function System SizeOf Gives the storage byte size of a type or variable
23890: Function System Slice Creates a slice of an array as an Open Array parameter
23891: Type System SmallInt An Integer type supporting values from -32768 to 32767
23892: Function System Sqr Gives the square of a number
23893: Function System Sqrt Gives the square root of a number
23894: Procedure System Str Converts an integer or floating point number to a string
23895: Type System String A data type that holds a string of characters
23896: Function System StringOfChar Creates a string with one character repeated many times
23897: Function SysUtils StringReplace Replace one or more substrings found within a string
23898: Function System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
23899: Function SysUtils StrScan Searches for a specific character in a constant string
23900: Function SysUtils StrToCurr Convert a number string into a currency value
23901: Function SysUtils StrToDate Converts a date string into a TDateTime value
23902: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
23903: Function SysUtils StrToFloat Convert a number string into a floating point value
23904: Function SysUtils StrToInt Convert an integer string into an Integer value
23905: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
23906: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
23907: Function SysUtils StrToIntDef Convert a string into an Integer value with default
23908: Function SysUtils StrToTime Converts a time string into a TDateTime value
23909: Function StrUtils StuffString Replaces a part of one string with another
23910: Function System Succ Increment an ordinal variable
23911: Function Math Sum Return the sum of an array of floating point values
23912:
23913: T
23914: Function Math Tan The Tangent of a number
23915: Type Classes TBits An object that can hold an infinite number of Boolean values
23916: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
23917: Type ConvUtils TConvType Defines a measurement type as used by Convert
23918: Type System TDateTime Data type holding a date and time value
23919: Type System Text Defines a file as a text file
23920: Type System TextFile Declares a file type for storing lines of text
23921: Type SysUtils TFloatFormat Formats for use in floating point number display functions
23922: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
23923: Keyword Then Part of an if statement - starts the true clause
23924: Variable SysUtils ThousandSeparator The character used to display the thousands separator
23925: Keyword ThreadVar Defines variables that are given separate instances per thread
23926: Function SysUtils Time Gives the current time
23927: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
23928: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
23929: Variable SysUtils TimeSeparator The character used to separate display time fields
23930: Function SysUtils TimeToStr Converts a TDateTime time value to a string
23931: Type Classes TList General purpose container of a list of objects
23932: Keyword To Prefixes an incremental for loop target value
23933: Type System TObject The base class type that is ancestor to all other classes
23934: Function DateUtils Tomorrow Gives the date tomorrow
23935: Type Dialogs TOpenDialog Displays a file selection dialog
23936: Type Types TPoint Holds X and Y integer values
23937: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
23938: Type Types TRect Holds rectangle coordinate values
23939: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
23940: Function SysUtils Trim Removes leading and trailing blanks from a string
23941: Function SysUtils TrimLeft Removes leading blanks from a string
23942: Function SysUtils TrimRight Removes trailing blanks from a string
23943: Function System Trunc The integer part of a floating point number
23944: Procedure System Truncate Truncates a file size - removes all data after the current position
23945: Keyword Try Starts code that has error trapping
23946: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
23947: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
23948: Type Classes TStringList Holds a variable length list of strings
23949: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
23950: Type System TThreadFunc Defines the function to be called by BeginThread
23951: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
23952: Keyword Type Defines a new category of variable or process
23953:
23954:
23955: U
23956: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
23957: Keyword Unit Defines the start of a unit file - a Delphi module
23958: Keyword Until Ends a Repeat control loop
23959: Function System UpCase Convert a Char value to upper case
23960: Function SysUtils UpperCase Change lower case characters in a string to upper case
23961: Keyword Uses Declares a list of Units to be imported
23962:
23963: V
23964: Procedure System Val Converts number strings to integer and floating point values
23965: Keyword Var Starts the definition of a section of data variables
23966: Type System Variant A variable type that can hold changing data types
23967: Function Variants VarType Gives the current type of a Variant variable
23968: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
23969: Directive Virtual Allows a class method to be overriden in derived classes
23970:
23971: W
23972: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
23973: Keyword While Repeat statements whilst a continuation condition is met
23974: Type System WideChar Variable type holding a single International character
23975: Function System WideCharToString Copies a null terminated WideChar string to a normal string
```

```
23976: Type System WideString A data type that holds a string of WideChars
23977: Keyword With A means of simplifying references to structured variables
23978: Type System Word An integer type supporting values 0 to 65535
23979: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
23980: Procedure System Write Write data to a binary or text file
23981: Procedure System WriteLn Write a complete line of data to a text file
23982:
23983: X
23984: Compiler Directive $X Controls some Pascal extension handling
23985: Keyword Xor Boolean Xor or bitwise Xor of two arguments
23986:
23987: Y
23988: Compiler Directive $Y Determines whether application symbol information is built
23989: Function DateUtils Yesterday Gives the date yesterday
23990:
23991: Z
23992: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
23993:
23994: ------------------------------------------------------------------------------
23995: mapX:
23996:  if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
23997:     writeln('cologne map found');
23998:     GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
23999:     writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
24000:     OpenMapX('church trier');
24001:     GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
24002:     writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));}
24003:     >>>     //latitude: '50.94133705'  longitude: '6.95812076100766'
24004:     // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
24005:     CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
24006:     Function SendInput( cInputs : UINT; var pInputs : TInput; cbSize : Integer) : UINT');
24007:     Function GetLastInputInfo( var plii : TLastInputInfo) : BOOL');
24008:
24009:
24010: maXbox Ref:
24011: Signature:
24012: SHA1:  maXbox3.exe 3EFB12E5729BDAA745373C2743F9DD1E93C9295D
24013: CRC32: maXbox3.exe 9435E377
24014:
24015: Ref:
24016:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
24017:     2. shdig: TSHA1Digest;
24018:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
24019:                for i:= 0 to 19 do write(BytetoHex(shdig[i]));
24020:
24021:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
24022:
24023:
24024: https://www.virustotal.
     com/en/file/aa4a38fed63d66b0e4cdbf6642533a84fc0d70ef7f0398909eaa98ab543e5cc4/analysis/1417190705/
24025:
24026: VirusTotal metadata
24027:
24028: First submission 2014-11-28 16:05:05 UTC ( 7 minutes ago )
24029: Last submission 2014-11-28 16:05:05 UTC ( 7 minutes ago )
24030: File names  Sunprise mX4
24031: maXbox3.exe
24032: maxbox3_9.exe
24033:
24034: SHA256:  aa4a38fed63d66b0e4cdbf6642533a84fc0d70ef7f0398909eaa98ab543e5cc4
24035: File name:  maXbox3.exe
24036: Detection ratio:  0 / 56
24037: Analysis date:  2014-11-28 16:05:05 UTC ( 2 minutes ago )
24038:
24039: MD5 0cad3130b08ca9e4411c810ece12af6a
24040: SHA1 3efb12e5729bdaa745373c2743f9dd1e93c9295d
24041: SHA256 aa4a38fed63d66b0e4cdbf6642533a84fc0d70ef7f0398909eaa98ab543e5cc4
24042: ssdeep 393216:5SsQEl8iDOZWNXYxbsjjztTsfNEnqqSdQ:VQEGWNckoNY0
24043:
24044: authentihash bb825f992fc9401de0738ca4703c206e472ec2e1211bfc383417bcdc49e23b49
24045: imphash 995a4c7c553245e979876cb44c73c127
24046:
24047: File size 22.5 MB ( 23614464 bytes )
24048: File type Win32 EXE
24049: Magic literal
24050: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
24051:
24052: TrID  Windows ActiveX control (36.4%)
24053: Inno Setup installer (34.3%)
24054: InstallShield setup (13.4%)
24055: Win32 EXE PECompact compressed (generic) (13.0%)
24056: Win32 Executable (generic) (1.4%)
24057:
24058:
24059:  ExifTool file metadata
24060:  ------------------------
24061:
24062: SpecialBuild mX4
24063: CodeSize 18578944
```

```
24064: SubsystemVersion 4.0
24065: Comments reduce to the max
24066: LinkerVersion 2.25
24067: ImageVersion  0.0
24068: FileSubtype 0
24069: FileVersionNumber 3.9.9.120
24070: LanguageCode German (Swiss)
24071: FileFlagsMask 0x003f
24072: FileDescription maXbox Delphi VM
24073: CharacterSet Windows, Latin1
24074: InitializedDataSize 5034496
24075: FileOS Win32
24076: TimeStamp 2014:11:28 07:52:53+01:00
24077: MIMEType application/octet-stream
24078: LegalCopyright Free Pascal Script
24079: FileVersion 3.9.9.120
24080: SpeziellesBuild mX4 Compiler Engine
24081: FileType Win32 EXE
24082: PEType PE32
24083: InternalName Sunprise mX4
24084: FileAccessDate 2014:11:28 17:05:21+01:00
24085: ProductVersion 3.9 Solar mX4
24086: UninitializedDataSize 0
24087: OSVersion 4.0
24088: FileCreateDate 2014:11:28 17:05:21+01:00
24089: OriginalFilename maxbox3_9.exe
24090: Subsystem Windows GUI
24091: MachineType Intel 386 or later, and compatibles
24092: CompanyName kleiner kommunikation
24093: LegalTrademarks maXbox
24094: ProductName maXbox
24095: ProductVersionNumber 3.9.9.120
24096: EntryPoint 0x11b8bc8
24097: ObjectFileType Executable application
24098:
24099:   ExifTool file metadata
24100:  -------------------------
24101:
24102: Developer metadata
24103:
24104: Publisher kleiner kommunikation
24105: Product maXbox
24106: Original name maxbox3_9.exe
24107: Internal name Sunprise mX4
24108: File version 3.9.9.120
24109: Description maXbox Delphi VM
24110: Comments reduce to the max
24111:
24112: PE header basic information
24113:
24114: Target machine Intel 386 or later processors and compatible processors
24115: Compilation timestamp 2014-11-28 06:52:53
24116: Link date 7:52 AM 11/28/2014
24117: Entry Point 0x011B8BC8
24118: Number of sections 10
24119:
24120: 10 PE sections
24121:
24122: Name Virtual address Virtual size Raw size Entropy MD5
24123: .text 4096 18530188 18530304 6.60 f713bdfd017af0db7502220027e1db91
24124: .itext 18534400 48396 48640 6.59 0f19c2a354db1e4670cc176da08c275f
24125: .data 18583552 263428 263680 5.58 1d26b778a838680bbe58d1b5cc1a885e
24126: .bss 18849792 381132 0 0.00 d41d8cd98f00b204e9800998ecf8427e
24127: .idata 19234816 55120 55296 5.63 f260089b89816d6b51bf67b81f21618f
24128: .edata 19292160 77 512 0.89 d750ae0f6bc96f2f43ca0d749817d42b
24129: .tls 19296256 208 0 0.00 d41d8cd98f00b204e9800998ecf8427e
24130: .rdata 19300352 24 512 0.28 182f0f96d2fd255ef9f76ec92bb0c3ff
24131: .reloc 19304448 1179672 1180160 6.76 b5abed6a26b4f7619aa1943694465635
24132: .rsrc 20488192 3534336 3534336 5.28 12861f6285d9b9dd393469215742faff
24133:
24134: PE imports
24135:
24136: PE exports CreateIncome
24137:
24138:  Number of PE resources by type
24139: RT_BITMAP 810
24140: RT_STRING 304
24141: RT_RCDATA 158
24142: RT_ICON 49
24143: RT_GROUP_ICON 43
24144: RT_CURSOR 31
24145: RT_GROUP_CURSOR 26
24146: WAVE 9
24147: RT_DIALOG 2
24148: RT_HTML 2
24149: RT_MESSAGETABLE 1
24150: RT_MANIFEST 1
24151: RT_VERSION 1
24152:
```

```
24153:
24154:  PE imports
24155:
24156: [+] AVICAP32.DLL  [+] AVICAP32.dll [+] GLU32.dll
24157: [+] IMAGEHLP.DLL  [+] MSVCRT.DLL   [+] MSVFW32.DLL
24158: [+] OpenGL32.dll  [+] SHFolder.dll [+] URLMON.DLL
24159: [+] advapi32.dll  [+] comctl32.dll [+] comdlg32.dll
24160: [+] gdi32.dll     [+] imagehlp.dll [+] imm32.dll
24161: [+] iphlpapi.dll  [+] kernel32.dll [+] mpr.dll
24162: [+] msacm32.dll   [+] ole32.dll    [+] oleacc.dll
24163: [+] oleaut32.dll  [+] oledlg.dll   [+] opengl32.dll
24164: [+] shell32.dll   [+] shlwapi.dll  [+] user32.dll
24165: [+] usp10.dll     [+] version.dll  [+] winhttp.dll
24166: [+] wininet.dll   [+] winmm.dll    [+] winspool.drv
24167: [+] ws2_32.dll    [+] wsock32.dll  [+] msimg32.dll
24168:
24169: Ref:
24170: https://www.virustotal.
       com/en/file/aa4a38fed63d66b0e4cdbf6642533a84fc0d70ef7f0398909eaa98ab543e5cc4/analysis/1417190705/
24171:
24172: ****************************************************************
24173: Release Notes maXbox 3.9.9.120 December 2014 CODEsign
24174: ****************************************************************
24175: Add 10 Units, 1Slides, NeuralNetwork, Pan3D View, GDIBackend
24176: 1043 unit uPSI_NeuralNetwork;
24177: 1044 unit uPSI_StExpr;
24178: 1045 unit uPSI_GR32_Geometry;
24179: 1046 unit uPSI_GR32_Containers;
24180: 1047 unit uPSI_GR32_Backends_VCL,
24181: 1048 unit uPSI_StSaturn; //Venus+Pluto+Mars+Merc+JupSat+++
24182: 1049 unit uPSI_JclParseUses;
24183: 1050 unit uPSI_JvFinalize;
24184: 1051 unit uPSI_panUnit1;
24185: 1052 unit uPSI_DD83u1;  //Arduino Tester
24186:
24187: SHA1:  maXbox3.exe 3EFB12E5729BDAA745373C2743F9DD1E93C9295D
24188: CRC32: maXbox3.exe 9435E377
24189:
24190:   ---- bigbitbox code_cleared_checked----
```