```
 1: ********************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9.160
 3: ********************************************************************
 4:
 5: ////////////////////////////////////////////////////////////////////////
 6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX3
 7: --------------------------------------------------------------------
 8:
 9: File Size of EXE: 23860224 V3.9.9.160 Jan 2015 To EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: ***************Now the Funclist*****************************************
11: Funclist Function : 14075 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 8641  //6792 //6401 4752
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1401 //995 //
16: def head:max: maXbox7: 09.01.2015 22:11:55
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: 24117! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 24575
22: ASize of EXE: 23860224 (23614464) (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.160: 8B4D7070BA40BDE17EEDBE78121BB8B474A1D6CF
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;//FileCreate2(FileName:string;Rights:Integer):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 MaxCalcS( const Formula : string): string');
1892: function MD5(const fileName: string): string;
1893: Function Mean( const Data : array of Double) : Extended
1894: Function Median( const X : TDynFloatArray) : Float
1895: Function Memory : Pointer
1896: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1897: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1898: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1899: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1900: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1901: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1902: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1903: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1904: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1905: Function MibToId( Mib : string) : string
1906: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1907: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1908: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1909: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1910: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1911: Procedure GetMidiOutputs( const List : TStrings)
1912: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
1913: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
1914: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1915: Function MIDINoteToStr( Note : TMIDINote) : string
1916: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1917: Procedure GetMidiOutputs( const List : TStrings)
1918: Procedure MidiOutCheck( Code : MMResult)
1919: Procedure MidiInCheck( Code : MMResult)
1920: Function MilliSecondOf( const AValue : TDateTime) : Word
1921: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1922: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1923: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1924: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1925: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
```

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

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

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

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

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

```
2357: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2358: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2359: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2360: Function RunByteCode(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2361: Function RunCompiledScript2(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2362: Function S_AddBackSlash( const ADirName : string) : string
2363: Function S_AllTrim( const cStr : string) : string
2364: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2365: Function S_Cut( const cStr : string; const iLen : integer) : string
2366: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2367: Function S_DirExists( const ADir : string) : Boolean
2368: Function S_Empty( const cStr : string) : boolean
2369: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2370: Function S_LargeFontsActive : Boolean
2371: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2372: Function S_LTrim( const cStr : string) : string
2373: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2374: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2375: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2376: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2377: Function S_RTrim( const cStr : string) : string
2378: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2379: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2380: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2381: Function S_Space( const iLen : integer) : String
2382: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2383: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2384: Function S_StrCRC32( const Text : string) : LongWORD
2385: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2386: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2387: Function S_StringtoUTF_8( const AString : string) : string
2388: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2389: function S_StrToReal(const cStr: string; var R: Double): Boolean
2390: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2391: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2392: Function S_UTF_8ToString( const AString : string) : string
2393: Function S_WBox( const AText : string) : integer
2394: Function SameDate( const A, B : TDateTime) : Boolean
2395: function SameDate(const A, B: TDateTime): Boolean;
2396: Function SameDateTime( const A, B : TDateTime) : Boolean
2397: function SameDateTime(const A, B: TDateTime): Boolean;
2398: Function SameFileName( S1, S2 : string) : Boolean
2399: Function SameText( S1, S2 : string) : Boolean
2400: function SameText(const S1: string; const S2: string): Boolean)
2401: Function SameTime( const A, B : TDateTime) : Boolean
2402: function SameTime(const A, B: TDateTime): Boolean;
2403: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2404: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2405: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2406: Function SampleVariance( const X : TDynFloatArray) : Float
2407: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2408: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2409: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2410: Function SaveToFile( const AFileName : TFileName) : Boolean
2411: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2412: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2413: Function ScanF(const aformat: String; const args: array of const): string;
2414: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2415: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2416: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2417: function SearchRecattr: integer;
2418: function SearchRecExcludeAttr: integer;
2419: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2420: function SearchRecname: string;
2421: function SearchRecsize: integer;
2422: function SearchRecTime: integer;
2423: Function Sec( const X : Extended) : Extended
2424: Function Secant( const X : Extended) : Extended
2425: Function SecH( const X : Extended) : Extended
2426: Function SecondOf( const AValue : TDateTime) : Word
2427: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2428: Function SecondOfTheHour( const AValue : TDateTime) : Word
2429: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2430: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2431: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2432: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2433: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2434: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2435: Function SectionExists( const Section : string) : Boolean
2436: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2437: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2438: function Seek(Offset:LongInt;Origin:Word):LongInt
2439: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2440: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2441: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2442: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
```

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

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

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

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

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

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

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

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

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

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

```
3314: Procedure _DBError( const Msg : string)
3315:  Const('TrueExpr','String '0=0
3316:  Const('sdfStandard16','String '''"''mm''/''dd''/''yyyy''"''
3317:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3318:  Const('sdfOracle','String '"TO_DATE(''"dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3319:  Const('sdfInterbase','String '"CAST(''"mm"/"dd"/"yyyy"'' AS DATE)"
3320:  Const('sdfMSSQL','String '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3321:  AddTypeS('Largeint', 'Longint
3322:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3323:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3324:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3325:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3326:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3327: (*----------------------------------------------------------------------------*)
3328: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3329: begin
3330:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3331:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3332:  Function JIniReadString( const FileName, Section, Line : string) : string
3333:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3334:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3335:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3336:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3337:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3338: end;
3339:
3340: (* === compile-time registration functions === *)
3341: (*----------------------------------------------------------------------------*)
3342: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3343: begin
3344:  'UnixTimeStart','LongInt'( 25569);
3345:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3346:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3347:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3348:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3349:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3350:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3351:  Function DayOfDate( const DateTime : TDateTime) : Integer
3352:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3353:  Function YearOfDate( const DateTime : TDateTime) : Integer
3354:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3355:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3356:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3357:  Function HourOfTime( const DateTime : TDateTime) : Integer
3358:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3359:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3360:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3361:  Function IsISOLongYear( const Year : Word) : Boolean;
3362:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3363:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3364:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3365:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3366:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3367:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3368:  Function JIsLeapYear( const Year : Integer) : Boolean;
3369:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3370:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3371:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3372:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3373:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3374:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3375:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3376:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3377:  Function HoursToMSecs( Hours : Integer) : Integer
3378:  Function MinutesToMSecs( Minutes : Integer) : Integer
3379:  Function SecondsToMSecs( Seconds : Integer) : Integer
3380:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3381:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3382:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3383:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3384:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3385:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3386:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3387:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3388:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3389:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3390:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3391:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3392:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3393:  Function DosDateTimeToStr( DateTime : Integer) : string
3394:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3395:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3396:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3397:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3398:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3399:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3400:  Function FileTimeToStr( const FileTime : TFileTime) : string
3401:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3402:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
```

```
3403:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3404:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3405:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3406:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3407:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3408:   TJclUnixTime32', 'Longword
3409:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3410:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3411:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3412:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3413:  Function JNullStamp : TTimeStamp
3414:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3415:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3416:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3417:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3418:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3419:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3420:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3421:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3422:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3423:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3424:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3425:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3426:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3427:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3428:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3429:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3430:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3431:   FindClass('TOBJECT'),'EJclDateTimeError
3432: end;
3433:
3434: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3435: begin
3436:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3437:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3438:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3439:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3440:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3441:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3442:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3443:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3444:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3445:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3446:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3447:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3448:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3449:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3450:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3451:  Function AbortShutDown : Boolean;
3452:  Function AbortShutDown1( const MachineName : string) : Boolean;
3453:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3454:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3455:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3456:   FindClass('TOBJECT'),'EJclCreateProcessError
3457:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3458:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
     Environment:PChar);
3459:   // with Add(EJclCreateProcessError) do
3460:  end;
3461:
3462:
3463: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3464: begin
3465:   //'AnsiSigns','Set').SetSet(['-', '+']);
3466:  'C1_UPPER','LongWord( $0001);
3467:  'C1_LOWER','LongWord( $0002);
3468:  'C1_DIGIT','LongWord').SetUInt( $0004);
3469:  'C1_SPACE','LongWord').SetUInt( $0008);
3470:  'C1_PUNCT','LongWord').SetUInt( $0010);
3471:  'C1_CNTRL','LongWord').SetUInt( $0020);
3472:  'C1_BLANK','LongWord').SetUInt( $0040);
3473:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3474:  'C1_ALPHA','LongWord').SetUInt( $0100);
3475:   AnsiChar', 'Char
3476:  Function StrIsAlpha( const S : AnsiString) : Boolean
3477:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3478:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3479:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3480:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3481:  Function StrIsDigit( const S : AnsiString) : Boolean
3482:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3483:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3484:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3485:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3486:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3487:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3488:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3489:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3490:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
```

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

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

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

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

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

```
{ RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
function RBTag(Parent: TWinControl): Integer;
{ AppMinimized returns True, if Application is minimized }
function AppMinimized: Boolean;
{ MessageBox is Application.MessageBox with string (not PChar) parameters.
  if Caption parameter = '', it replaced with Application.Title }
function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
  Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
  Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
{ Delay stop program execution to MSec msec }
procedure Delay(MSec: Longword);
procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
procedure EnableControls(Control: TWinControl; const Enable: Boolean);
procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
function PanelBorder(Panel: TCustomPanel): Integer;
function Pixels(Control: TControl; APixels: Integer): Integer;
procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
procedure Error(const Msg: string);
procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
  {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean): string;
function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean): Integer;
function ItemHtPlain(const Text: string): string;
{ ClearList - clears list of TObject }
procedure ClearList(List: TList);
procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
{ RTTI support }
function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
function GetPropStr(Obj: TObject; const PropName: string): string;
function GetPropOrd(Obj: TObject; const PropName: string): Integer;
function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
procedure PrepareIniSection(SS: TStrings);
{ following functions are not documented because they are don't work properly, so don't use them }
{$IFDEF COMPILER2}
function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
{$ENDIF}

procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
begin
 Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
 Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
 Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
 Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
 Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
 Procedure BoxSetItem( List : TWinControl; Index : Integer)
 Function BoxGetFirstSelection( List : TWinControl) : Integer
 Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
end;

procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
begin
 Const('MaxInitStrNum','LongInt'( 9);
Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
  : array of AnsiString; MaxSplit : Integer) : Integer
Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
  string; MaxSplit : Integer) : Integer
Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
  QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
Function JvAnsiStrStrip( S : AnsiString) : AnsiString
Function JvStrStrip( S : string) : string
Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
Function StrEatWhiteSpace( const S : string) : string
Function HexToAscii( const S : AnsiString) : AnsiString
Function AsciiToHex( const S : AnsiString) : AnsiString
Function StripQuotes( const S1 : AnsiString) : AnsiString
Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
Function HexPCharToInt( S1 : PAnsiChar) : Integer
Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
Function JvValidIdentifier( S1 : String) : Boolean
Function JvEndChar( X : AnsiChar) : Boolean
Procedure JvGetToken( S1, S2 : PAnsiChar)
Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
Function IsKeyword( S1 : PAnsiChar) : Boolean
Function JvValidVarReference( S1 : PAnsiChar) : Boolean
Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
```

```
4010: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
4011: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
4012: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
4013: Function GetTokenCount : Integer
4014: Procedure ResetTokenCount
4015: end;
4016:
4017: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
4018: begin
4019:   SIRegister_TJvQueryParamsDialog(CL);
4020:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
4021: end;
4022:
4023: ******************************** JvStrUtil /  JvStrUtils;****************************
4024: function FindNotBlankCharPos(const S: string): Integer;
4025: function AnsiChangeCase(const S: string): string;
4026: function GetWordOnPos(const S: string; const P: Integer): string;
4027: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4028: function Cmp(const S1, S2: string): Boolean;
4029: { Spaces returns string consists on N space chars }
4030: function Spaces(const N: Integer): string;
4031: { HasChar returns True, if char, Ch, contains in string, S }
4032: function HasChar(const Ch: Char; const S: string): Boolean;
4033: function HasAnyChar(const Chars: string; const S: string): Boolean;
4034: { SubStr returns substring from string, S, separated with Separator string}
4035: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4036: { SubStrEnd same to previous function but Index numerated from the end of string }
4037: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4038: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4039: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
4040: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4041: { GetXYByPos is same to previous function, but returns X position in line too}
4042: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4043: { AddSlash returns string with added slash char to Dir parameter, if needed }
4044: function AddSlash2(const Dir: TFileName): string;
4045: { AddPath returns FileName with Path, if FileName not contain any path }
4046: function AddPath(const FileName, Path: TFileName): TFileName;
4047: { ExePath returns ExtractFilePath(ParamStr(0)) }
4048: function ExePath: TFileName;
4049: function LoadTextFile(const FileName: TFileName): string;
4050: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4051: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4052: function ConcatSep(const S, S2, Separator: string): string;
4053: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4054: function FileEquMask(FileName, Mask: TFileName): Boolean;
4055: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4056:   Masks must be separated with comma (';') }
4057: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4058: function StringEndsWith(const Str, SubStr: string): Boolean;
4059: function ExtractFilePath2(const FileName: string): string;
4060: function StrToOem(const AnsiStr: string): string;
4061: { StrToOem translates a string from the Windows character set into the OEM character set. }
4062: function OemToAnsiStr(const OemStr: string): string;
4063: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4064: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4065: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4066: function ReplaceStr(const S, Srch, Replace: string): string;
4067: { Returns string with every occurrence of Srch string replaced with Replace string. }
4068: function DelSpace(const S: string): string;
4069: { DelSpace return a string with all white spaces removed. }
4070: function DelChars(const S: string; Chr: Char): string;
4071: { DelChars return a string with all Chr characters removed. }
4072: function DelBSpace(const S: string): string;
4073: { DelBSpace trims leading spaces from the given string. }
4074: function DelESpace(const S: string): string;
4075: { DelESpace trims trailing spaces from the given string. }
4076: function DelRSpace(const S: string): string;
4077: { DelRSpace trims leading and trailing spaces from the given string. }
4078: function DelSpace1(const S: string): string;
4079: { DelSpace1 return a string with all non-single white spaces removed. }
4080: function Tab2Space(const S: string; Numb: Byte): string;
4081: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4082: function NPos(const C: string; S: string; N: Integer): Integer;
4083: { NPos searches for a N-th position of substring C in a given string. }
4084: function MakeStr(C: Char; N: Integer): string;
4085: function MS(C: Char; N: Integer): string;
4086: { MakeStr return a string of length N filled with character C. }
4087: function AddChar(C: Char; const S: string; N: Integer): string;
4088: { AddChar return a string left-padded to length N with characters C. }
4089: function AddCharR(C: Char; const S: string; N: Integer): string;
4090: { AddCharR return a string right-padded to length N with characters C. }
4091: function LeftStr(const S: string; N: Integer): string;
4092: { LeftStr return a string right-padded to length N with blanks. }
4093: function RightStr(const S: string; N: Integer): string;
4094: { RightStr return a string left-padded to length N with blanks. }
4095: function CenterStr(const S: string; Len: Integer): string;
4096: { CenterStr centers the characters in the string based upon the Len specified. }
4097: function CompStr(const S1, S2: string): Integer;
4098: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
```

```
4099: function CompText(const S1, S2: string): Integer;
4100: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4101: function Copy2Symb(const S: string; Symb: Char): string;
4102: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4103: function Copy2SymbDel(var S: string; Symb: Char): string;
4104: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4105: function Copy2Space(const S: string): string;
4106: { Copy2Symb returns a substring of a string S from begining to first white space. }
4107: function Copy2SpaceDel(var S: string): string;
4108: { Copy2SpaceDel returns a substring of a string S from begining to first
4109:   white space and removes this substring from S. }
4110: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4111: { Returns string, with the first letter of each word in uppercase,
4112:   all other letters in lowercase. Words are delimited by WordDelims. }
4113: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4114: { WordCount given a set of word delimiters, returns number of words in S. }
4115: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4116: { Given a set of word delimiters, returns start position of N'th word in S. }
4117: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4118: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4119: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4120: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4121:   delimiters, return the N'th word in S. }
4122: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4123: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4124: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4125: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4126: function QuotedString(const S: string; Quote: Char): string;
4127: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4128: function ExtractQuotedString(const S: string; Quote: Char): string;
4129: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4130:    and reduces pairs of Quote characters within the quoted string to a single character. }
4131: function FindPart(const HelpWilds, InputStr: string): Integer;
4132: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4133: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4134: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4135: function XorString(const Key, Src: ShortString): ShortString;
4136: function XorEncode(const Key, Source: string): string;
4137: function XorDecode(const Key, Source: string): string;
4138: { ** Command line routines ** }
4139: {$IFNDEF COMPILER4_UP}
4140: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4141: {$ENDIF}
4142: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4143: { ** Numeric string handling routines ** }
4144: function Numb2USA(const S: string): string;
4145: { Numb2USA converts numeric string S to USA-format. }
4146: function Dec2Hex(N: Longint; A: Byte): string;
4147: function D2H(N: Longint; A: Byte): string;
4148: { Dec2Hex converts the given value to a hexadecimal string representation
4149:   with the minimum number of digits (A) specified. }
4150: function Hex2Dec(const S: string): Longint;
4151: function H2D(const S: string): Longint;
4152: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4153: function Dec2Numb(N: Longint; A, B: Byte): string;
4154: { Dec2Numb converts the given value to a string representation with the
4155:   base equal to B and with the minimum number of digits (A) specified. }
4156: function Numb2Dec(S: string; B: Byte): Longint;
4157: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4158: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4159: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4160: function IntToRoman(Value: Longint): string;
4161: { IntToRoman converts the given value to a roman numeric string representation. }
4162: function RomanToInt(const S: string): Longint;
4163: { RomanToInt converts the given string to an integer value. If the string
4164:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4165: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4166: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4167: ********************************* JvFileUtil;*********************************
4168: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4169: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4170: procedure MoveFile(const FileName, DestName: TFileName);
4171: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4172: {$IFDEF COMPILER4_UP}
4173: function GetFileSize(const FileName: string): Int64;
4174: {$ELSE}
4175: function GetFileSize(const FileName: string): Longint;
4176: {$ENDIF}
4177: function FileDateTime(const FileName: string): TDateTime;
4178: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4179: function DeleteFiles(const FileMask: string): Boolean;
4180: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4181: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4182: function NormalDir(const DirName: string): string;
4183: function RemoveBackSlash(const DirName: string): string;
4184: function ValidFileName(const FileName: string): Boolean;
4185: function DirExists(Name: string): Boolean;
4186: procedure ForceDirectories(Dir: string);
```

```
4187: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4188:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4189: {$IFDEF COMPILER4_UP}
4190: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4191: {$ENDIF}
4192: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4193: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4194: {$IFDEF COMPILER4_UP}
4195: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4196: {$ENDIF}
4197: function GetTempDir: string;
4198: function GetWindowsDir: string;
4199: function GetSystemDir: string;
4200: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4201: {$IFDEF WIN32}
4202: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4203: function ShortToLongFileName(const ShortName: string): string;
4204: function ShortToLongPath(const ShortName: string): string;
4205: function LongToShortFileName(const LongName: string): string;
4206: function LongToShortPath(const LongName: string): string;
4207: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4208: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4209: {$ENDIF WIN32}
4210: {$IFNDEF COMPILER3_UP}
4211: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4212: {$ENDIF}
4213: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4214: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4215: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4216: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4217:
4218: //**************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4219:  Procedure VariantClear( var V : Variant)
4220:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4221:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4222:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4223:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4224:  Procedure VariantSub( const src : Variant; var dst : Variant)
4225:  Procedure VariantMul( const src : Variant; var dst : Variant)
4226:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4227:  Procedure VariantMod( const src : Variant; var dst : Variant)
4228:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4229:  Procedure VariantOr( const src : Variant; var dst : Variant)
4230:  Procedure VariantXor( const src : Variant; var dst : Variant)
4231:  Procedure VariantShl( const src : Variant; var dst : Variant)
4232:  Procedure VariantShr( const src : Variant; var dst : Variant)
4233:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4234:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4235:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4236:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4237:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4238:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4239:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4240:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4241:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4242:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4243:  Function VariantNot( const V1 : Variant) : Variant
4244:  Function VariantNeg( const V1 : Variant) : Variant
4245:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4246:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4247:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4248:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4249:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4250:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4251:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4252:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4253:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4254:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4255: end;
4256:
4257: ************************************unit uPSI_JvgUtils;************************************
4258: function IsEven(I: Integer): Boolean;
4259: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4260: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4261: procedure SwapInt(var I1, I2: Integer);
4262: function Spaces(Count: Integer): string;
4263: function DupStr(const Str: string; Count: Integer): string;
4264: function DupChar(C: Char; Count: Integer): string;
4265: procedure Msg(const AMsg: string);
4266: function RectW(R: TRect): Integer;
4267: function RectH(R: TRect): Integer;
4268: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4269: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4270: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4271: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4272:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4273: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4274: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4275:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
```

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

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

```
4451: {$ENDIF MSWINDOWS}
4452: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4453: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4454: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4455: procedure SaveString(const AFile, AText: string);
4456: Procedure SaveStringasFile( const AFile, AText : string)
4457: function LoadStringJ(const AFile: string): string;
4458: Function LoadStringofFile( const AFile : string) : string
4459: Procedure SaveStringtoFile( const AFile, AText : string)
4460: Function LoadStringfromFile( const AFile : string) : string
4461: function HexToColor(const AText: string): TColor;
4462: function UppercaseHTMLTags(const AText: string): string;
4463: function LowercaseHTMLTags(const AText: string): string;
4464: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4465: function RelativePath(const ASrc, ADst: string): string;
4466: function GetToken(var Start: Integer; const SourceText: string): string;
4467: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4468: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4469: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4470: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4471: // parses the beginning of an attribute: space + alpha character
4472: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4473: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4474: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4475: // parses all name=value attributes to the attributes TStringList
4476: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4477: // checks if a name="value" pair exists and returns any value
4478: function GetStrValue(const AText, AName, ADefault: string): string;
4479: // retrieves string value from a line like:
4480: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4481: // returns ADefault when not found
4482: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4483: // same for a color
4484: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4485: // same for an Integer
4486: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4487: // same for a float
4488: function GetBoolValue(const AText, AName: string): Boolean;
4489: // same for Boolean but without default
4490: function GetValue(const AText, AName: string): string;
4491: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4492: procedure SetValue(var AText: string; const AName, AValue: string);
4493: // sets a string value in a line
4494: procedure DeleteValue(var AText: string; const AName: string);
4495: // deletes a AName="value" pair from AText
4496: procedure GetNames(AText: string; AList: TStringList);
4497: // get a list of names from a string with name="value" pairs
4498: function GetHTMLColor(AColor: TColor): string;
4499: // converts a color value to the HTML hex value
4500: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4501: // finds a string backward case sensitive
4502: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4503: // finds a string backward case insensitive
4504: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4505:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4506: // finds a text range, e.g. <TD>....</TD> case sensitive
4507: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4508:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4509: // finds a text range, e.g. <TD>....</td> case insensitive
4510: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4511:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4512: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4513: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4514:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4515: // finds a text range backward, e.g. <TD>....</td> case insensitive
4516: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4517:   var RangeEnd: Integer): Boolean;
4518: // finds a HTML or XML tag:  <....>
4519: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4520:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4521: // finds the innertext between opening and closing tags
4522: function Easter(NYear: Integer): TDateTime;
4523: // returns the easter date of a year.
4524: function GetWeekNumber(Today: TDateTime): string;
4525: //gets a datecode. Returns year and weeknumber in format: YYWW
4526: function ParseNumber(const S: string): Integer;
4527: // parse number returns the last position, starting from 1
4528: function ParseDate(const S: string): Integer;
4529: // parse a SQL style data string from positions 1,
4530: // starts and ends with #
4531:
4532: ****************************************unit JvJCLUtils;****************************************
4533:
4534: function VarIsInt(Value: Variant): Boolean;
4535:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4536: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4537: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4538: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4539: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
```

```
4540: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4541: function GetWordOnPos(const S: string; const P: Integer): string;
4542: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4543: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4544: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4545: { GetWordOnPosEx working like GetWordOnPos function, but
4546:   also returns Word position in iBeg, iEnd variables }
4547: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4548: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4549: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4550: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4551: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4552: { GetEndPosCaret returns the caret position of the last char. For the position
4553:   after the last char of Text you must add 1 to the returned X value. }
4554: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4555: { GetEndPosCaret returns the caret position of the last char. For the position
4556:   after the last char of Text you must add 1 to the returned X value. }
4557: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4558: function SubStrBySeparator(const S:string;const Index:Integer;const
      Separator:string;StartIndex:Int=1):string;
4559: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
      Separator:WideString;StartIndex:Int:WideString;
4560: { SubStrEnd same to previous function but Index numerated from the end of string }
4561: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4562: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4563: function SubWord(P: PChar; var P2: PChar): string;
4564: function CurrencyByWord(Value: Currency): string;
4565: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4566: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4567: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4568: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4569: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4570: { ReplaceString searches for all substrings, OldPattern,
4571:   in a string, S, and replaces them with NewPattern }
4572: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4573: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
      WideString;StartIndex:Integer=1):WideString;
4574: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4575: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4576: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4577: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4578:
4579: { Next 4 function for russian chars transliterating.
4580:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4581: procedure Dos2Win(var S: AnsiString);
4582: procedure Win2Dos(var S: AnsiString);
4583: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4584: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4585: function Win2Koi(const S: AnsiString): AnsiString;
4586: { FillString fills the string Buffer with Count Chars }
4587: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4588: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4589: { MoveString copies Count Chars from Source to Dest }
4590: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4591: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4592:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4593: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4594: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4595: { MoveWideChar copies Count WideChars from Source to Dest }
4596: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4597: { FillNativeChar fills Buffer with Count NativeChars }
4598: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4599: { MoveWideChar copies Count WideChars from Source to Dest }
4600: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4601: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4602: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4603: { Spaces returns string consists on N space chars }
4604: function Spaces(const N: Integer): string;
4605: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4606: function AddSpaces(const S: string; const N: Integer): string;
4607: function SpacesW(const N: Integer): WideString;
4608: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4609: { function LastDateRUS for russian users only }
4610: { returns date relative to current date: 'ääà äiý íàçàä' }
4611: function LastDateRUS(const Dat: TDateTime): string;
4612: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4613: function CurrencyToStr(const Cur: Currency): string;
4614: { HasChar returns True, if Char, Ch, contains in string, S }
4615: function HasChar(const Ch: Char; const S: string): Boolean;
4616: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4617: function HasAnyChar(const Chars: string; const S: string): Boolean;
4618: {$IFNDEF COMPILER12_UP}
4619: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
```

```
4620: {$ENDIF ~COMPILER12_UP}
4621: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4622: function CountOfChar(const Ch: Char; const S: string): Integer;
4623: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4624: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4625: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4626: function StrPosW(S, SubStr: PWideChar): PWideChar;
4627: function StrLenW(S: PWideChar): Integer;
4628: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4629: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4630: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4631: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4632: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4633:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4634: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4635: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4636: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4637: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4638: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4639: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4640: Function ScreenPixelFormat : TPixelFormat
4641: Function ScreenColorCount : Integer
4642: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4643: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4644: //  SIRegister_TJvGradient(CL);
4645:
4646: {************************************ files routines}
4647: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4648: const
4649:   {$IFDEF MSWINDOWS}
4650:   DefaultCaseSensitivity = False;
4651:   {$ENDIF MSWINDOWS}
4652:   {$IFDEF UNIX}
4653:   DefaultCaseSensitivity = True;
4654:   {$ENDIF UNIX}
4655: { GenTempFileName returns temporary file name on
4656:   drive, there FileName is placed }
4657: function GenTempFileName(FileName: string): string;
4658: { GenTempFileNameExt same to previous function, but
4659:   returning filename has given extension, FileExt }
4660: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4661: { ClearDir clears folder Dir }
4662: function ClearDir(const Dir: string): Boolean;
4663: { DeleteDir clears and than delete folder Dir }
4664: function DeleteDir(const Dir: string): Boolean;
4665: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4666: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4667: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4668:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4669: function FileEquMasks(FileName, Masks: TFileName;
4670:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4671: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4672: {$IFDEF MSWINDOWS}
4673: { LZFileExpand expand file, FileSource,
4674:   into FileDest. Given file must be compressed, using MS Compress program }
4675: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4676: {$ENDIF MSWINDOWS}
4677: { FileGetInfo fills SearchRec record for specified file attributes}
4678: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4679: { HasSubFolder returns True, if folder APath contains other folders }
4680: function HasSubFolder(APath: TFileName): Boolean;
4681: { IsEmptyFolder returns True, if there are no files or
4682:   folders in given folder, APath}
4683: function IsEmptyFolder(APath: TFileName): Boolean;
4684: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4685: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4686: { AddPath returns FileName with Path, if FileName not contain any path }
4687: function AddPath(const FileName, Path: TFileName): TFileName;
4688: function AddPaths(const PathList, Path: string): string;
4689: function ParentPath(const Path: TFileName): TFileName;
4690: function FindInPath(const FileName, PathList: string): TFileName;
4691: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4692: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4693: { HasParam returns True, if program running with specified parameter, Param }
4694: function HasParam(const Param: string): Boolean;
4695: function HasSwitch(const Param: string): Boolean;
4696: function Switch(const Param: string): string;
4697: { ExePath returns ExtractFilePath(ParamStr(0)) }
4698: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4699: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4700: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4701: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4702: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4703: {**** Graphic routines }
4704: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4705: function IsTTFontSelected(const DC: HDC): Boolean;
```

```
4706: function KeyPressed(VK: Integer): Boolean;
4707: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4708: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4709: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4710: {**** Color routines }
4711: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4712: function RGBToBGR(Value: Cardinal): Cardinal;
4713: //function ColorToPrettyName(Value: TColor): string;
4714: //function PrettyNameToColor(const Value: string): TColor;
4715: {**** other routines }
4716: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4717: function IntPower(Base, Exponent: Integer): Integer;
4718: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4719: function StrToBool(const S: string): Boolean;
4720: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4721: function VarToInt(V: Variant): Integer;
4722: function VarToFloat(V: Variant): Double;
4723: { following functions are not documented because they not work properly sometimes, so do not use them }
4724: // (rom) ReplaceStrings1, GetSubStr removed
4725: function GetLongFileName(const FileName: string): string;
4726: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4727: function GetParameter: string;
4728: function GetComputerID: string;
4729: function GetComputerName: string;
4730: {**** string routines }
4731: { ReplaceAllStrings searches for all substrings, Words,
4732:   in a string, S, and replaces them with Frases with the same Index. }
4733: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4734: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4735:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with the
     same Index, and then update NewSelStart variable }
4736: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4737: { CountOfLines calculates the lines count in a string, S,
4738:   each line must be separated from another with CrLf sequence }
4739: function CountOfLines(const S: string): Integer;
4740: { DeleteLines deletes all lines from strings which in the words,  words.
4741:   The word of will be deleted from strings. }
4742: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4743: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4744:   Lines contained only spaces also deletes. }
4745: procedure DeleteEmptyLines(Ss: TStrings);
4746: { SQLAddWhere addes or modifies existing where-statement, where,
4747:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4748:   it must be started on the begining of any line }
4749: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4750: {**** files routines - }
4751: {$IFDEF MSWINDOWS}
4752: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4753:   Resource can be compressed using MS Compress program}
4754: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4755: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
     Boolean;
4756: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4757: {$ENDIF MSWINDOWS}
4758: { IniReadSection read section, Section, from ini-file,
4759:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4760:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4761: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4762: { LoadTextFile load text file, FileName, into string }
4763: function LoadTextFile(const FileName: TFileName): string;
4764: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4765: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4766: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4767: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4768: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4769: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4770: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4771: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4772: { RATextCalcHeight calculate needed height for
4773:   correct output, using RATextOut or RATextOutEx functions }
4774: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4775: { Cinema draws some visual effect }
4776: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4777: { Roughed fills rect with special 3D pattern }
4778: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4779: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
     and height, AWidth, AHeight and placed on a specified Index, Index in the  source bitmap }
4780: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4781: { TextWidth calculate text with for writing using standard desktop font }
4782: function TextWidth(const AStr: string): Integer;
4783: { TextHeight calculate text height for writing using standard desktop font }
4784: function TextHeight(const AStr: string): Integer;
4785: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4786: procedure Error(const Msg: string);
4787: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4788:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4789: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4790: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4791:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
```

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

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

```
4969: { Dec2Hex converts the given value to a hexadecimal string representation
4970:   with the minimum number of digits (A) specified. }
4971: function Hex2Dec(const S: string): Longint;
4972: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4973: function Dec2Numb(N: Int64; A, B: Byte): string;
4974: { Dec2Numb converts the given value to a string representation with the
4975:   base equal to B and with the minimum number of digits (A) specified. }
4976: function Numb2Dec(S: string; B: Byte): Int64;
4977: { Numb2Dec converts the given B-based numeric string to the corresponding
4978:   integer value. }
4979: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4980: { IntToBin converts the given value to a binary string representation
4981:   with the minimum number of digits specified. }
4982: function IntToRoman(Value: Longint): string;
4983: { IntToRoman converts the given value to a roman numeric string representation. }
4984: function RomanToInt(const S: string): Longint;
4985: { RomanToInt converts the given string to an integer value. If the string
4986:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4987: function FindNotBlankCharPos(const S: string): Integer;
4988: function FindNotBlankCharPosW(const S: WideString): Integer;
4989: function AnsiChangeCase(const S: string): string;
4990: function WideChangeCase(const S: string): string;
4991: function StartsText(const SubStr, S: string): Boolean;
4992: function EndsText(const SubStr, S: string): Boolean;
4993: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4994: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4995: {end JvStrUtils}
4996: {$IFDEF UNIX}
4997: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4998: {$ENDIF UNIX}
4999: { begin JvFileUtil }
5000: function FileDateTime(const FileName: string): TDateTime;
5001: function HasAttr(const FileName: string; Attr: Integer): Boolean;
5002: function DeleteFilesEx(const FileMasks: array of string): Boolean;
5003: function NormalDir(const DirName: string): string;
5004: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
5005: function ValidFileName(const FileName: string): Boolean;
5006: {$IFDEF MSWINDOWS}
5007: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5008: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5009: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5010: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5011: {$ENDIF MSWINDOWS}
5012: function GetWindowsDir: string;
5013: function GetSystemDir: string;
5014: function ShortToLongFileName(const ShortName: string): string;
5015: function LongToShortFileName(const LongName: string): string;
5016: function ShortToLongPath(const ShortName: string): string;
5017: function LongToShortPath(const LongName: string): string;
5018: {$IFDEF MSWINDOWS}
5019: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
5020: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5021: {$ENDIF MSWINDOWS}
5022: { end JvFileUtil }
5023: // Works like PtInRect but includes all edges in comparision
5024: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5025: // Works like PtInRect but excludes all edges from comparision
5026: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5027: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5028: function IsFourDigitYear: Boolean;
5029: { moved from JvJVCLUTils }
5030: //Open an object with the shell (url or something like that)
5031: function OpenObject(const Value: string): Boolean; overload;
5032: function OpenObject(Value: PChar): Boolean; overload;
5033: {$IFDEF MSWINDOWS}
5034: //Raise the last Exception
5035: procedure RaiseLastWin32; overload;
5036: procedure RaiseLastWin32(const Text: string); overload;
5037: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
5038: function GetFileVersion(const AFileName: string): Cardinal;
5039: {$EXTERNALSYM GetFileVersion}
5040: //Get version of Shell.dll
5041: function GetShellVersion: Cardinal;
5042: {$EXTERNALSYM GetShellVersion}
5043: // CD functions  on HW
5044: procedure OpenCdDrive;
5045: procedure CloseCdDrive;
5046: // returns True if Drive is accessible
5047: function DiskInDrive(Drive: Char): Boolean;
5048: {$ENDIF MSWINDOWS}
5049: //Same as linux function ;)
5050: procedure PError(const Text: string);
5051: // execute a program without waiting
5052: procedure Exec(const FileName, Parameters, Directory: string);
5053: // execute a program and wait for it to finish
5054: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5055: // returns True if this is the first instance of the program that is running
```

```
5056: function FirstInstance(const ATitle: string): Boolean;
5057: // restores a window based on it's classname and Caption. Either can be left empty
5058: // to widen the search
5059: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5060: // manipulate the traybar and start button
5061: procedure HideTraybar;
5062: procedure ShowTraybar;
5063: procedure ShowStartButton(Visible: Boolean = True);
5064: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5065: procedure MonitorOn;
5066: procedure MonitorOff;
5067: procedure LowPower;
5068: // send a key to the window named AppName
5069: function SendKey(const AppName: string; Key: Char): Boolean;
5070: {$IFDEF MSWINDOWS}
5071: // returns a list of all win currently visible, the Objects property is filled with their window handle
5072: procedure GetVisibleWindows(List: TStrings);
5073: Function GetVisibleWindowsF( List : TStrings):TStrings');
5074: // associates an extension to a specific program
5075: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5076: procedure AddToRecentDocs(const FileName: string);
5077: function GetRecentDocs: TStringList;
5078: {$ENDIF MSWINDOWS}
5079: function CharIsMoney(const Ch: Char): Boolean;
5080: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5081: function IntToExtended(I: Integer): Extended;
5082: { GetChangedText works out the new text given the current cursor pos & the key pressed
5083:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5084: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5085: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5086: //function StrIsInteger(const S: string): Boolean;
5087: function StrIsFloatMoney(const Ps: string): Boolean;
5088: function StrIsDateTime(const Ps: string): Boolean;
5089: function PreformatDateString(Ps: string): string;
5090: function BooleanToInteger(const B: Boolean): Integer;
5091: function StringToBoolean(const Ps: string): Boolean;
5092: function SafeStrToDateTime(const Ps: string): TDateTime;
5093: function SafeStrToDate(const Ps: string): TDateTime;
5094: function SafeStrToTime(const Ps: string): TDateTime;
5095: function StrDelete(const psSub, psMain: string): string;
5096:   { returns the fractional value of pcValue}
5097: function TimeOnly(pcValue: TDateTime): TTime;
5098: { returns the integral value of pcValue }
5099: function DateOnly(pcValue: TDateTime): TDate;
5100: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5101: const { TDateTime value used to signify Null value}
5102:   NullEquivalentDate: TDateTime = 0.0;
5103: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5104: // Replacement for Win32Check to avoid platform specific warnings in D6
5105: function OSCheck(RetVal: Boolean): Boolean;
5106: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5107:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5108:   not be forced to use FileCtrl unnecessarily }
5109: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5110: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5111: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5112: function MinimizeString(const S: string; const MaxLen: Integer): string;
5113: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
5114: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
     minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
     found.}
5115: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5116: {$ENDIF MSWINDOWS}
5117: procedure ResourceNotFound(ResID: PChar);
5118: function EmptyRect: TRect;
5119: function RectWidth(R: TRect): Integer;
5120: function RectHeight(R: TRect): Integer;
5121: function CompareRect(const R1, R2: TRect): Boolean;
5122: procedure RectNormalize(var R: TRect);
5123: function RectIsSquare(const R: TRect): Boolean;
5124: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5125: //If AMaxSize = -1 ,then auto calc Square's max size
5126: {$IFDEF MSWINDOWS}
5127: procedure FreeUnusedOle;
5128: function GetWindowsVersion: string;
5129: function LoadDLL(const LibName: string): THandle;
5130: function RegisterServer(const ModuleName: string): Boolean;
5131: function UnregisterServer(const ModuleName: string): Boolean;
5132: {$ENDIF MSWINDOWS}
5133: { String routines }
5134: function GetEnvVar(const VarName: string): string;
5135: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5136: function StringToPChar(var S: string): PChar;
5137: function StrPAlloc(const S: string): PChar;
5138: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5139: function DropT(const S: string): string;
5140: { Memory routines }
5141: function AllocMemo(Size: Longint): Pointer;
```

```
5142: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5143: procedure FreeMemo(var fpBlock: Pointer);
5144: function GetMemoSize(fpBlock: Pointer): Longint;
5145: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5146: { Manipulate huge pointers routines }
5147: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5148: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5149: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5150: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5151: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5152: function WindowClassName(Wnd: THandle): string;
5153: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5154: procedure ActivateWindow(Wnd: THandle);
5155: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5156: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5157: { SetWindowTop put window to top without recreating window }
5158: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5159: procedure CenterWindow(Wnd: THandle);
5160: function MakeVariant(const Values: array of Variant): Variant;
5161: { Convert dialog units to pixels and backwards }
5162: {$IFDEF MSWINDOWS}
5163: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5164: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5165: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5166: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5167: {$ENDIF MSWINDOWS}
5168: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5169: {$IFDEF BCB}
5170: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5171: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5172: {$ELSE}
5173: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5174: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5175: {$ENDIF BCB}
5176: {$IFDEF MSWINDOWS}
5177: { BrowseForFolderNative displays Browse For Folder dialog }
5178: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5179: {$ENDIF MSWINDOWS}
5180: procedure AntiAlias(Clip: TBitmap);
5181: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5182: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5183:   ABitmap: TBitmap; const SourceRect: TRect);
5184: function IsTrueType(const FontName: string): Boolean;
5185: // Removes all non-numeric characters from AValue and returns the resulting string
5186: function TextToValText(const AValue: string): string;
5187: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5188: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5189: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5190: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5191: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5192:
5193: ****************************************************unit uPSI_JvTFUtils;
5194:  Function JExtractYear( ADate : TDateTime) : Word
5195:  Function JExtractMonth( ADate : TDateTime) : Word
5196:  Function JExtractDay( ADate : TDateTime) : Word
5197:  Function ExtractHours( ATime : TDateTime) : Word
5198:  Function ExtractMins( ATime : TDateTime) : Word
5199:  Function ExtractSecs( ATime : TDateTime) : Word
5200:  Function ExtractMSecs( ATime : TDateTime) : Word
5201:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5202:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5203:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5204:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5205:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5206:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5207:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5208:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5209:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5210:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5211:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5212:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5213:  Procedure EnsureMonth( Month : Word)
5214:  Procedure EnsureDOW( DOW : Word)
5215:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5216:  Function Lesser( N1, N2 : Integer) : Integer
5217:  Function Greater( N1, N2 : Integer) : Integer
5218:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5219:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5220:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5221:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5222:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5223:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5224:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
     AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5225:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
     HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5226:  Function JRectWidth( ARect : TRect) : Integer
5227:  Function JRectHeight( ARect : TRect) : Integer
```

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

```
5317:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5318:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5319:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5320:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5321:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5322:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5323:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5324:  'ESBLnPi','Extended').setExtended( 1.1447298858494001741 4);
5325:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5326:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5327:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5328:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5329:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5330:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5331:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5332:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5333:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5334:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5335:  'ESBPiToE','Extended').setExtended( 22.459157183610454 73);
5336:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5337:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5338:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5339:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5340:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5341:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5342:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5343:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5344:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5345:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5346:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5347:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5348:   //LongWord', 'Cardinal
5349:   TBitList', 'Word
5350:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5351:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5352:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5353:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5354:  Function SameFloat( const X1, X2 : Extended) : Boolean
5355:  Function FloatIsZero( const X : Extended) : Boolean
5356:  Function FloatIsPositive( const X : Extended) : Boolean
5357:  Function FloatIsNegative( const X : Extended) : Boolean
5358:  Procedure IncLim( var B : Byte; const Limit : Byte)
5359:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5360:  Procedure IncLimW( var B : Word; const Limit : Word)
5361:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5362:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5363:  Procedure DecLim( var B : Byte; const Limit : Byte)
5364:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5365:  Procedure DecLimW( var B : Word; const Limit : Word)
5366:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5367:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5368:  Function MaxB( const B1, B2 : Byte) : Byte
5369:  Function MinB( const B1, B2 : Byte) : Byte
5370:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5371:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5372:  Function MaxW( const B1, B2 : Word) : Word
5373:  Function MinW( const B1, B2 : Word) : Word
5374:  Function esbMaxI( const B1, B2 : Integer) : Integer
5375:  Function esbMinI( const B1, B2 : Integer) : Integer
5376:  Function MaxL( const B1, B2 : LongInt) : LongInt
5377:  Function MinL( const B1, B2 : LongInt) : LongInt
5378:  Procedure SwapB( var B1, B2 : Byte)
5379:  Procedure SwapSI( var B1, B2 : ShortInt)
5380:  Procedure SwapW( var B1, B2 : Word)
5381:  Procedure SwapI( var B1, B2 : SmallInt)
5382:  Procedure SwapL( var B1, B2 : LongInt)
5383:  Procedure SwapI32( var B1, B2 : Integer)
5384:  Procedure SwapC( var B1, B2 : LongWord)
5385:  Procedure SwapInt64( var X, Y : Int64)
5386:  Function esbSign( const B : LongInt) : ShortInt
5387:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5388:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5389:  Function Max3Word( const X1, X2, X3 : Word) : Word
5390:  Function Min3Word( const X1, X2, X3 : Word) : Word
5391:  Function MaxBArray( const B : array of Byte) : Byte
5392:  Function MaxWArray( const B : array of Word) : Word
5393:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5394:  Function MaxIArray( const B : array of Integer) : Integer
5395:  Function MaxLArray( const B : array of LongInt) : LongInt
5396:  Function MinBArray( const B : array of Byte) : Byte
5397:  Function MinWArray( const B : array of Word) : Word
5398:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5399:  Function MinIArray( const B : array of Integer) : Integer
5400:  Function MinLArray( const B : array of LongInt) : LongInt
5401:  Function SumBArray( const B : array of Byte) : Byte
5402:  Function SumBArray2( const B : array of Byte) : Word
5403:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5404:  Function SumSIArray2( const B : array of ShortInt) : Integer
5405:  Function SumWArray( const B : array of Word) : Word
```

```
5406:  Function SumWArray2( const B : array of Word) : LongInt
5407:  Function SumIArray( const B : array of Integer) : Integer
5408:  Function SumLArray( const B : array of LongInt) : LongInt
5409:  Function SumLWArray( const B : array of LongWord) : LongWord
5410:  Function ESBDigits( const X : LongWord) : Byte
5411:  Function BitsHighest( const X : LongWord) : Integer
5412:  Function ESBBitsNeeded( const X : LongWord) : Integer
5413:  Function esbGCD( const X, Y : LongWord) : LongWord
5414:  Function esbLCM( const X, Y : LongInt) : Int64
5415:  //Function esbLCM( const X, Y : LongInt) : LongInt
5416:  Function RelativePrime( const X, Y : LongWord) : Boolean
5417:  Function Get87ControlWord : TBitList
5418:  Procedure Set87ControlWord( const CWord : TBitList)
5419:  Procedure SwapExt( var X, Y : Extended)
5420:  Procedure SwapDbl( var X, Y : Double)
5421:  Procedure SwapSing( var X, Y : Single)
5422:  Function esbSgn( const X : Extended) : ShortInt
5423:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5424:  Function ExtMod( const X, Y : Extended) : Extended
5425:  Function ExtRem( const X, Y : Extended) : Extended
5426:  Function CompMOD( const X, Y : Comp) : Comp
5427:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5428:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5429:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5430:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5431:  Function MaxExt( const X, Y : Extended) : Extended
5432:  Function MinExt( const X, Y : Extended) : Extended
5433:  Function MaxEArray( const B : array of Extended) : Extended
5434:  Function MinEArray( const B : array of Extended) : Extended
5435:  Function MaxSArray( const B : array of Single) : Single
5436:  Function MinSArray( const B : array of Single) : Single
5437:  Function MaxCArray( const B : array of Comp) : Comp
5438:  Function MinCArray( const B : array of Comp) : Comp
5439:  Function SumSArray( const B : array of Single) : Single
5440:  Function SumEArray( const B : array of Extended) : Extended
5441:  Function SumSqEArray( const B : array of Extended) : Extended
5442:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5443:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5444:  Function SumCArray( const B : array of Comp) : Comp
5445:  Function FactorialX( A : LongWord) : Extended
5446:  Function PermutationX( N, R : LongWord) : Extended
5447:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5448:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5449:  Function esbGeometricMean( const X : array of Extended) : Extended
5450:  Function esbHarmonicMean( const X : array of Extended) : Extended
5451:  Function ESBMean( const X : array of Extended) : Extended
5452:  Function esbSampleVariance( const X : array of Extended) : Extended
5453:  Function esbPopulationVariance( const X : array of Extended) : Extended
5454:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5455:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5456:  Function GetMedian( const SortedX : array of Extended) : Extended
5457:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5458:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5459:  Function ESBMagnitude( const X : Extended) : Integer
5460:  Function ESBTan( Angle : Extended) : Extended
5461:  Function ESBCot( Angle : Extended) : Extended
5462:  Function ESBCosec( const Angle : Extended) : Extended
5463:  Function ESBSec( const Angle : Extended) : Extended
5464:  Function ESBArcTan( X, Y : Extended) : Extended
5465:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5466:  Function ESBArcCos( const X : Extended) : Extended
5467:  Function ESBArcSin( const X : Extended) : Extended
5468:  Function ESBArcSec( const X : Extended) : Extended
5469:  Function ESBArcCosec( const X : Extended) : Extended
5470:  Function ESBLog10( const X : Extended) : Extended
5471:  Function ESBLog2( const X : Extended) : Extended
5472:  Function ESBLogBase( const X, Base : Extended) : Extended
5473:  Function Pow2( const X : Extended) : Extended
5474:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5475:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5476:  Function XtoY( const X, Y : Extended) : Extended
5477:  Function esbTenToY( const Y : Extended) : Extended
5478:  Function esbTwoToY( const Y : Extended) : Extended
5479:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5480:  Function esbISqrt( const I : LongWord) : Longword
5481:  Function ILog2( const I : LongWord) : LongWord
5482:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5483:  Function ESBArCosh( X : Extended) : Extended
5484:  Function ESBArSinh( X : Extended) : Extended
5485:  Function ESBArTanh( X : Extended) : Extended
5486:  Function ESBCosh( X : Extended) : Extended
5487:  Function ESBSinh( X : Extended) : Extended
5488:  Function ESBTanh( X : Extended) : Extended
5489:  Function InverseGamma( const X : Extended) : Extended
5490:  Function esbGamma( const X : Extended) : Extended
5491:  Function esbLnGamma( const X : Extended) : Extended
5492:  Function esbBeta( const X, Y : Extended) : Extended
5493:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5494: end;
```

```
5495:
5496: ******************************Integer Huge Cardinal Utils*************************
5497: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5498: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5499: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5500: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5501: Function BitCount_8( Value : byte) : integer
5502: Function BitCount_16( Value : uint16) : integer
5503: Function BitCount_32( Value : uint32) : integer
5504: Function BitCount_64( Value : uint64) : integer
5505: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5506:  Procedure ( CountPrimalityTests : integer)
5507:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5508:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5509:  Function isCoPrime( a, b : THugeCardinal) : boolean
5510:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5511:  Function hasSmallFactor( p : THugeCardinal) : boolean
5512:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5513:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5514:  Const('StandardExponent','LongInt'( 65537);
5515: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5516:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5517:
5518: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5519: begin
5520:   AddTypeS('TXRTLInteger', 'array of Integer
5521:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5522:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5523:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5524:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5525:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5526:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5527:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5528:   'BitsPerByte','LongInt'( 8);
5529:   BitsPerDigit','LongInt'( 32);
5530:   SignBitMask','LongWord( $80000000);
5531:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5532:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5533:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5534:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5535:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5536:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5537:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5538:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5539:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5540:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5541:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5542:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5543:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5544:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5545:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5546:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5547:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5548:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5549:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5550:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5551:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5552:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5553:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5554:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5555:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5556:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5557:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5558:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5559:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5560:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5561:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5562:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5563:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5564:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5565:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5566:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5567:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5568:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5569:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5570:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5571:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5572:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5573:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5574:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5575:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5576:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5577:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
```

```
5578:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5579:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5580:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5581:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5582:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5583:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5584:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5585:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5586:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5587:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5588:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5589:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5590:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5591:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5592:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5593:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5594:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5595:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5596:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5597:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5598:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5599:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5600:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5601:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5602:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5603:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5604:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5605: end;
5606:
5607: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5608: begin
5609:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5610:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5611:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5612:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5613:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5614:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5615:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5616:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5617:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5618:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5619:  Procedure JvXPPlaceText( const AParent: TControl; const ACanvas : TCanvas; const AText : TCaption; const
      AFont : TFont; const AEnabled,AShowAccelChar:Boolean;const AAlignment:TAlignment;const
      AWordWrap:Boolean;var Rect:TRect)
5620: end;
5621:
5622:
5623: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5624: begin
5625:  Function StrDec( S : String) : String
5626:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5627:  Function ReadNumFromEdit( Edit : TEdit) : Float
5628:  Procedure WriteNumToFile( var F : Text; X : Float)
5629: end;
5630:
5631: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5632: begin
5633:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5634:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5635:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5636:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5637:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5638:  Procedure TeX_SetGraphTitle( Title : String)
5639:  Procedure TeX_SetOxTitle( Title : String)
5640:  Procedure TeX_SetOyTitle( Title : String)
5641:  Procedure TeX_PlotOxAxis
5642:  Procedure TeX_PlotOyAxis
5643:  Procedure TeX_PlotGrid( Grid : TGrid)
5644:  Procedure TeX_WriteGraphTitle
5645:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5646:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5647:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5648:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5649:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5650:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5651:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5652:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5653:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5654:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5655:  Function Xcm( X : Float) : Float
5656:  Function Ycm( Y : Float) : Float
5657: end;
5658:
5659: *---------------------------------------------------------------------------*)
```

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

```
5744:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5745:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5746:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5747:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5748:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5749:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
       SubString:ShortString):Bool;
5750:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
       SubStr:ShortString):Bool;
5751:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
       SubString:ShortString):Bool;
5752:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
       SubString:ShortString):Bool;
5753:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5754:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5755:  Function ExtractTokensS(const S,
       Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5756:  Function IsChAlphaS( C : Char) : Boolean
5757:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5758:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5759:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5760:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5761:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5762:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5763:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5764:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5765:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5766:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5767:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
       Cardinal):ShortString;
5768:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
       Replacements:Cardinal):ShortString;
5769:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5770:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
       Replacements:Cardinal):ShortString
5771:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
       Replacements:Cardinal):ShortString
5772:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5773:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5774:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5775:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5776: end;
5777:
5778:
5779: ********unit uPSI_StUtils; from Systools4*****************************************************
5780: Function SignL( L : LongInt) : Integer
5781: Function SignF( F : Extended) : Integer
5782: Function MinWord( A, B : Word) : Word
5783: Function MidWord( W1, W2, W3 : Word) : Word
5784: Function MaxWord( A, B : Word) : Word
5785: Function MinLong( A, B : LongInt) : LongInt
5786: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5787: Function MaxLong( A, B : LongInt) : LongInt
5788: Function MinFloat( F1, F2 : Extended) : Extended
5789: Function MidFloat( F1, F2, F3 : Extended) : Extended
5790: Function MaxFloat( F1, F2 : Extended) : Extended
5791: Function MakeInteger16( H, L : Byte) : SmallInt
5792: Function MakeWordS( H, L : Byte) : Word
5793: Function SwapNibble( B : Byte) : Byte
5794: Function SwapWord( L : LongInt) : LongInt
5795: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5796: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5797: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5798: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5799: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5800: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5801: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5802: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5803: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5804: Procedure ExchangeBytes( var I, J : Byte)
5805: Procedure ExchangeWords( var I, J : Word)
5806: Procedure ExchangeLongInts( var I, J : LongInt)
5807: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5808: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5809: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5810: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5811: //*****************uPSI_StFIN;************************************************************
5812: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5813: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
       Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5814: Function BondDuration( Settlement,Maturity:TStDate;Rate,
       Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5815: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
       Extended
5816: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5817: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5818: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
```

```
5819: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5820: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5821: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5822: Function DollarToDecimalText( DecDollar : Extended) : string
5823: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5824: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5825: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5826: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5827: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5828: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5829: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5830: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5831: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5832: Function IsCardValid( const S : string) : Boolean
5833: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5834: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5835: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5836: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5837: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5838: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5839: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5840: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5841: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5842: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5843: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5844: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5845: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5846: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5847: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5848: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5849: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5850: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5851: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5852: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5853:
5854: //*****************************************unit uPSI_StAstroP;
5855: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5856: //*****unit unit uPSI_StStat; Statistic Package of SysTools***************************************
5857: Function AveDev( const Data : array of Double) : Double
5858: Function AveDev16( const Data, NData : Integer) : Double
5859: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5860: Function Correlation( const Data1, Data2 : array of Double) : Double
5861: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5862: Function Covariance( const Data1, Data2 : array of Double) : Double
5863: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5864: Function DevSq( const Data : array of Double) : Double
5865: Function DevSq16( const Data, NData : Integer) : Double
5866: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5867:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5868: Function GeometricMeanS( const Data : array of Double) : Double
5869: Function GeometricMean16( const Data, NData : Integer) : Double
5870: Function HarmonicMeanS( const Data : array of Double) : Double
5871: Function HarmonicMean16( const Data, NData : Integer) : Double
5872: Function Largest( const Data : array of Double; K : Integer) : Double
5873: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5874: Function MedianS( const Data : array of Double) : Double
5875: Function Median16( const Data, NData : Integer) : Double
5876: Function Mode( const Data : array of Double) : Double
5877: Function Mode16( const Data, NData : Integer) : Double
5878: Function Percentile( const Data : array of Double; K : Double) : Double
5879: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5880: Function PercentRank( const Data : array of Double; X : Double) : Double
5881: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5882: Function Permutations( Number, NumberChosen : Integer) : Extended
5883: Function Combinations( Number, NumberChosen : Integer) : Extended
5884: Function FactorialS( N : Integer) : Extended
5885: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5886: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5887: Function Smallest( const Data : array of Double; K : Integer) : Double
5888: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5889: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5890: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5891:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5892:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5893: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5894: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5895: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5896: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5897: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5898: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
```

```
5899: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5900: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5901: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5902: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5903: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5904: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5905: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5906: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5907: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5908: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5909: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5910: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5911: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5912: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5913: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5914: Function NormSDist( Z : Single) : Single
5915: Function NormSInv( Probability : Single) : Single
5916: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5917: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5918: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5919: Function Erfc( X : Single) : Single
5920: Function GammaLn( X : Single) : Single
5921: Function LargestSort( const Data : array of Double; K : Integer) : Double
5922: Function SmallestSort( const Data : array of double; K : Integer) : Double
5923:
5924: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5925:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5926:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5927:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5928:  Function DefaultMergeName( MergeNum : Integer) : string
5929:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5930:
5931: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5932: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5933:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5934:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5935:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5936:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5937:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5938:  Function LunarPhase( UT : TStDateTimeRec) : Double
5939:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5940:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5941:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5942:  Function FullMoon( D : TStDate) : TStLunarRecord
5943:  Function LastQuarter( D : TStDate) : TStLunarRecord
5944:  Function NewMoon( D : TStDate) : TStLunarRecord
5945:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5946:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5947:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5948:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5949:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5950:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5951:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5952:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5953:  Function SiderealTime( UT : TStDateTimeRec) : Double
5954:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5955:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5956:  Function SEaster( Y, Epoch : Integer) : TStDate
5957:  Function DateTimeToAJD( D : TDateTime) : Double
5958:  Function HoursMin( RA : Double) : ShortString
5959:  Function DegsMin( DC : Double) : ShortString
5960:  Function AJDToDateTime( D : Double) : TDateTime
5961:
5962: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5963:  Function CurrentDate : TStDate
5964:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5965:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5966:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5967:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5968:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5969:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5970:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5971:  Function WeekOfYear( Julian : TStDate) : Byte
5972:  Function AstJulianDate( Julian : TStDate) : Double
5973:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5974:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5975:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5976:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5977:  Function StIsLeapYear( Year : Integer) : Boolean
5978:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5979:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5980:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5981:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5982:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5983:  Function CurrentTime : TStTime
5984:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5985:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5986:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5987:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
```

```
5988:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5989:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5990:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5991:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5992:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5993:  Function StDateToDateTime( D : TStDate) : TDateTime
5994:  Function StTimeToDateTime( T : TStTime) : TDateTime
5995:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5996:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5997:
5998: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5999: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
6000:  Function MonthToString( const Month : Integer) : string
6001:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
6002:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
6003:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
6004:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
6005:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
6006:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
6007:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
6008:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
6009:  Function TimeStringToStTime( const Picture, S : string) : TStTime
6010:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6011:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6012:  Function DateStringIsBlank( const Picture, S : string) : Boolean
6013:  Function InternationalDate( ForceCentury : Boolean) : string
6014:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
6015:  Function InternationalTime( ShowSeconds : Boolean) : string
6016:  Procedure ResetInternationalInfo
6017:
6018: procedure SIRegister_StBase(CL: TPSPascalCompiler);
6019:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
6020:  Function AnsiUpperCaseShort32( const S : string) : string
6021:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
6022:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
6023:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
6024:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
6025:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
6026:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
6027:  Function Upcase( C : AnsiChar) : AnsiChar
6028:  Function LoCase( C : AnsiChar) : AnsiChar
6029:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
6030:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
6031:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6032:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6033:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6034:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
6035:  Procedure RaiseContainerError( Code : longint)
6036:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6037:  Function ProductOverflow( A, B : LongInt) : Boolean
6038:  Function StNewStr( S : string) : PShortString
6039:  Procedure StDisposeStr( PS : PShortString)
6040:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
6041:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
6042:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6043:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6044:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6045:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6046:
6047: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6048: begin
6049:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6050:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6051:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6052:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6053:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6054: end;
6055:
6056: //**********unit unit ; StMath Package of SysTools*****************************************
6057: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6058: Function PowerS( Base, Exponent : Extended) : Extended
6059: Function StInvCos( X : Double) : Double
6060: Function StInvSin( Y : Double) : Double
6061: Function StInvTan2( X, Y : Double) : Double
6062: Function StTan( A : Double) : Double
6063: Procedure DumpException;  //unit StExpEng;
6064: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6065:
6066: //**********unit unit ; StCRC Package of SysTools*****************************************
6067: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6068: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6069: Function Adler32OfFile( FileName : AnsiString) : LongInt
6070: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6071: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6072: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6073: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6074: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6075: Function Crc32OfFile( FileName : AnsiString) : LongInt
6076: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
```

```
6077: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6078: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6079: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6080: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6081: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6082:
6083: //**********unit unit ; StBCD Package of SysTools*****************************************
6084: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6085: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6086: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6087: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6088: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6089: Function NegBcd( const B : TbcdS) : TbcdS
6090: Function AbsBcd( const B : TbcdS) : TbcdS
6091: Function FracBcd( const B : TbcdS) : TbcdS
6092: Function IntBcd( const B : TbcdS) : TbcdS
6093: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6094: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6095: Function ValBcd( const S : string) : TbcdS
6096: Function LongBcd( L : LongInt) : TbcdS
6097: Function ExtBcd( E : Extended) : TbcdS
6098: Function ExpBcd( const B : TbcdS) : TbcdS
6099: Function LnBcd( const B : TbcdS) : TbcdS
6100: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6101: Function PowBcd( const B, E : TbcdS) : TbcdS
6102: Function SqrtBcd( const B : TbcdS) : TbcdS
6103: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6104: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6105: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6106: Function IsIntBcd( const B : TbcdS) : Boolean
6107: Function TruncBcd( const B : TbcdS) : LongInt
6108: Function BcdExt( const B : TbcdS) : Extended
6109: Function RoundBcd( const B : TbcdS) : LongInt
6110: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6111: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6112: Function FormatBcd( const Format : string; const B : TbcdS) : string
6113: Function StrGeneralBcd( const B : TbcdS) : string
6114: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6115: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6116:
6117: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6118: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6119: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6120: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6121: Function StDeEscape( const EscStr : AnsiString) : Char
6122: Function StDoEscape( Delim : Char) : AnsiString
6123: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6124: Function AnsiHashText( const S : string; Size : Integer) : Integer
6125: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6126: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6127: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6128:
6129: //**********unit unit ; StNetCon Package of SysTools*****************************************
6130:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6131:     Constructor Create( AOwner : TComponent)
6132:     Function Connect : DWord
6133:     Function Disconnect : DWord
6134:     RegisterProperty('Password', 'String', iptrw);
6135:     Property('UserName', 'String', iptrw);
6136:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6137:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6138:     Property('LocalDevice', 'String', iptrw);
6139:     Property('ServerName', 'String', iptrw);
6140:     Property('ShareName', 'String', iptrw);
6141:     Property('OnConnect', 'TNotifyEvent', iptrw);
6142:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6143:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6144:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6145:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6146:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6147:   end;
6148: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6149: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6150: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6151: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6152: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6153: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6154: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6155: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6156: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6157: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6158: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6159: Function SuspendThread( hThread : THandle) : DWORD
6160: Function ResumeThread( hThread : THandle) : DWORD
6161: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6162: Function GetCurrentThread : THandle
6163: Procedure ExitThread( dwExitCode : DWORD)
6164: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
```

```
6165: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6166: Procedure EndThread(ExitCode: Integer);
6167: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6168: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6169: Procedure FreeProcInstance( Proc : FARPROC)
6170: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6171: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6172: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6173: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6174: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6175: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6176: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6177: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6178: Function CurrentParallelJobInfo : TParallelJobInfo
6179: Function ObtainParallelJobInfo : TParallelJobInfo
6180: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6181: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6182: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6183: Function
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL');
6184: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL');
6185: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD) : BOOL');
6186: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6187: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6188:
6189: ************************************************unit uPSI_JclMime;
6190:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6191:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6192:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6193:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6194:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6195:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6196:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6197:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6198:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6199:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6200:
6201: ************************************************unit uPSI_JclPrint;
6202:  Procedure DirectPrint( const Printer, Data : string)
6203:  Procedure SetPrinterPixelsPerInch
6204:  Function GetPrinterResolution : TPoint
6205:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6206:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6207:
6208:
6209: //***************************************unit uPSI_ShLwApi;***********************************
6210:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6211:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6212:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6213:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6214:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6215:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6216:  Function StrDup( lpSrch : PChar) : PChar
6217:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6218:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6219:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6220:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6221: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6222:  Function StrPBrk( psz, pszSet : PChar) : PChar
6223:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6224:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6225:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6226:  Function StrSpn( psz, pszSet : PChar) : Integer
6227:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6228:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6229:  Function StrToInt( lpSrch : PChar) : Integer
6230:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6231:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6232:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6233:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6234:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6235:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6236:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6237:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6238:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6239:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6240:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6241:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6242:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6243:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6244:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6245:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6246:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6247:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6248:  STIF_DEFAULT','LongWord( $00000000);
```

```
6249:  STIF_SUPPORT_HEX','LongWord( $00000001);
6250:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6251:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6252:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6253:  Function PathAddBackslash( pszPath : PChar) : PChar
6254:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6255:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6256:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6257:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6258:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6259:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6260:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6261:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6262:  Function PathFileExists( pszPath : PChar) : BOOL
6263:  Function PathFindExtension( pszPath : PChar) : PChar
6264:  Function PathFindFileName( pszPath : PChar) : PChar
6265:  Function PathFindNextComponent( pszPath : PChar) : PChar
6266:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6267:  Function PathGetArgs( pszPath : PChar) : PChar
6268:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6269:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6270:  Function PathGetCharType( ch : Char) : UINT
6271:  GCT_INVALID','LongWord( $0000);
6272:  GCT_LFNCHAR','LongWord( $0001);
6273:  GCT_SHORTCHAR','LongWord( $0002);
6274:  GCT_WILD','LongWord( $0004);
6275:  GCT_SEPARATOR','LongWord( $0008);
6276:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6277:  Function PathIsDirectory( pszPath : PChar) : BOOL
6278:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6279:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6280:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6281:  Function PathIsRelative( pszPath : PChar) : BOOL
6282:  Function PathIsRoot( pszPath : PChar) : BOOL
6283:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6284:  Function PathIsUNC( pszPath : PChar) : BOOL
6285:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6286:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6287:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6288:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6289:  Function PathIsURL( pszPath : PChar) : BOOL
6290:  Function PathMakePretty( pszPath : PChar) : BOOL
6291:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6292:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6293:  Procedure PathQuoteSpaces( lpsz : PChar)
6294:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6295:  Procedure PathRemoveArgs( pszPath : PChar)
6296:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6297:  Procedure PathRemoveBlanks( pszPath : PChar)
6298:  Procedure PathRemoveExtension( pszPath : PChar)
6299:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6300:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6301:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6302:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6303:  Function PathSkipRoot( pszPath : PChar) : PChar
6304:  Procedure PathStripPath( pszPath : PChar)
6305:  Function PathStripToRoot( pszPath : PChar) : BOOL
6306:  Procedure PathUnquoteSpaces( lpsz : PChar)
6307:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6308:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6309:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6310:  Procedure PathUndecorate( pszPath : PChar)
6311:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6312:  URL_SCHEME_INVALID','LongInt'( - 1);
6313:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6314:  URL_SCHEME_FTP','LongInt'( 1);
6315:  URL_SCHEME_HTTP','LongInt'( 2);
6316:  URL_SCHEME_GOPHER','LongInt'( 3);
6317:  URL_SCHEME_MAILTO','LongInt'( 4);
6318:  URL_SCHEME_NEWS','LongInt'( 5);
6319:  URL_SCHEME_NNTP','LongInt'( 6);
6320:  URL_SCHEME_TELNET','LongInt'( 7);
6321:  URL_SCHEME_WAIS','LongInt'( 8);
6322:  URL_SCHEME_FILE','LongInt'( 9);
6323:  URL_SCHEME_MK','LongInt'( 10);
6324:  URL_SCHEME_HTTPS','LongInt'( 11);
6325:  URL_SCHEME_SHELL','LongInt'( 12);
6326:  URL_SCHEME_SNEWS','LongInt'( 13);
6327:  URL_SCHEME_LOCAL','LongInt'( 14);
6328:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6329:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6330:  URL_SCHEME_ABOUT','LongInt'( 17);
6331:  URL_SCHEME_RES','LongInt'( 18);
6332:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6333:  URL_SCHEME', 'Integer
6334:  URL_PART_NONE','LongInt'( 0);
6335:  URL_PART_SCHEME','LongInt'( 1);
6336:  URL_PART_HOSTNAME','LongInt'( 2);
6337:  URL_PART_USERNAME','LongInt'( 3);
```

```
6338:  URL_PART_PASSWORD','LongInt'( 4);
6339:  URL_PART_PORT','LongInt'( 5);
6340:  URL_PART_QUERY','LongInt'( 6);
6341:  URL_PART', 'DWORD
6342:  URLIS_URL','LongInt'( 0);
6343:  URLIS_OPAQUE','LongInt'( 1);
6344:  URLIS_NOHISTORY','LongInt'( 2);
6345:  URLIS_FILEURL','LongInt'( 3);
6346:  URLIS_APPLIABLE','LongInt'( 4);
6347:  URLIS_DIRECTORY','LongInt'( 5);
6348:  URLIS_HASQUERY','LongInt'( 6);
6349:  TUrlIs', 'DWORD
6350:  URL_UNESCAPE','LongWord( $10000000);
6351:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6352:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6353:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6354:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6355:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6356:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6357:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6358:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6359:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6360:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6361:  URL_INTERNAL_PATH','LongWord( $00800000);
6362:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6363:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6364:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6365:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6366:  URL_APPLY_DEFAULT','LongWord( $00000001);
6367:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6368:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6369:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6370:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6371:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6372:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6373:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6374:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6375:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6376:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6377:  Function UrlGetLocation( psz1 : PChar) : PChar
6378:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6379:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6380:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6381:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6382:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6383:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6384:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6385:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6386:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6387:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6388:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6389:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6390:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6391:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6392:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6393:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6394:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6395:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6396:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6397:  SHREGDEL_DEFAULT','LongWord( $00000000);
6398:  SHREGDEL_HKCU','LongWord( $00000001);
6399:  SHREGDEL_HKLM','LongWord( $00000010);
6400:  SHREGDEL_BOTH','LongWord( $00000011);
6401:  SHREGENUM_DEFAULT','LongWord( $00000000);
6402:  SHREGENUM_HKCU','LongWord( $00000001);
6403:  SHREGENUM_HKLM','LongWord( $00000010);
6404:  SHREGENUM_BOTH','LongWord( $00000011);
6405:  SHREGSET_HKCU','LongWord( $00000001);
6406:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6407:  SHREGSET_HKLM','LongWord( $00000004);
6408:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6409:  TSHRegDelFlags', 'DWORD
6410:  TSHRegEnumFlags', 'DWORD
6411:  HUSKEY', 'THandle
6412:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6413:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6414:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6415:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6416:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6417:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6418:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6419:  ASSOCF_VERIFY','LongWord( $00000040);
6420:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6421:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6422:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6423:  ASSOCF', 'DWORD
6424:  ASSOCSTR_COMMAND','LongInt'( 1);
6425:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
```

```
6426:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6427:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6428:  ASSOCSTR_NOOPEN','LongInt'( 5);
6429:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6430:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6431:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6432:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6433:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6434:  ASSOCSTR_INFOTIP','LongInt'( 11);
6435:  ASSOCSTR_MAX','LongInt'( 12);
6436:  ASSOCSTR', 'DWORD
6437:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6438:  ASSOCKEY_APP','LongInt'( 2);
6439:  ASSOCKEY_CLASS','LongInt'( 3);
6440:  ASSOCKEY_BASECLASS','LongInt'( 4);
6441:  ASSOCKEY_MAX','LongInt'( 5);
6442:  ASSOCKEY', 'DWORD
6443:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6444:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6445:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6446:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6447:  ASSOCDATA_MAX','LongInt'( 5);
6448:  ASSOCDATA', 'DWORD
6449:  ASSOCENUM_NONE','LongInt'( 0);
6450:  ASSOCENUM', 'DWORD
6451:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6452:  SHACF_DEFAULT $00000000);
6453:  SHACF_FILESYSTEM','LongWord( $00000001);
6454:  SHACF_URLHISTORY','LongWord( $00000002);
6455:  SHACF_URLMRU','LongWord( $00000004);
6456:  SHACF_USETAB','LongWord( $00000008);
6457:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6458:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6459:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6460:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6461:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6462:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6463:  Procedure SHSetThreadRef( punk : IUnknown)
6464:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6465:  CTF_INSIST','LongWord( $00000001);
6466:  CTF_THREAD_REF','LongWord( $00000002);
6467:  CTF_PROCESS_REF','LongWord( $00000004);
6468:  CTF_COINIT','LongWord( $00000008);
6469:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6470:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6471:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6472:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6473:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6474:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6475:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6476:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6477:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6478:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6479:  Function SetRectEmpty( var lprc : TRect) : BOOL
6480:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6481:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6482:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6483:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6484:
6485:  Function InitializeFlatSB( hWnd : HWND) : Bool
6486:  Procedure UninitializeFlatSB( hWnd : HWND)
6487:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6488:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6489:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6490:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6491:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6492:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6493:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6494:
6495:
6496: //  ***************************************** 204 unit uPSI_ShellAPI;
6497: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6498: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6499: Procedure DragFinish( Drop : HDROP)
6500: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6501: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6502: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6503: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6504: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6505: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6506: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6507: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6508: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6509: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6510: Procedure SHFreeNameMappings( hNameMappings : THandle)
6511:
6512:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6513:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6514:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
```

```
6515:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6516:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6517:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6518:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6519:  Function SimpleXMLEncode( const S : string) : string
6520:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6521:  Function XMLEncode( const S : string) : string
6522:  Function XMLDecode( const S : string) : string
6523:  Function EntityEncode( const S : string) : string
6524:  Function EntityDecode( const S : string) : string
6525:
6526: procedure RIRegister_CPort_Routines(S: TPSExec);
6527:  Procedure EnumComPorts( Ports : TStrings)
6528:  Procedure ListComPorts( Ports : TStrings)
6529:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6530:  Function GetComPorts: TStringlist;
6531:  Function StrToBaudRate( Str : string) : TBaudRate
6532:  Function StrToStopBits( Str : string) : TStopBits
6533:  Function StrToDataBits( Str : string) : TDataBits
6534:  Function StrToParity( Str : string) : TParityBits
6535:  Function StrToFlowControl( Str : string) : TFlowControl
6536:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6537:  Function StopBitsToStr( StopBits : TStopBits) : string
6538:  Function DataBitsToStr( DataBits : TDataBits) : string
6539:  Function ParityToStr( Parity : TParityBits) : string
6540:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6541:  Function ComErrorsToStr( Errors : TComErrors) : String
6542:
6543:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6544:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6545:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6546:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6547:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6548:  Function GetMessagePos : DWORD
6549:  Function GetMessageTime : Longint
6550:  Function GetMessageExtraInfo : Longint
6551:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6552:  Procedure JAddToRecentDocs( const Filename : string)
6553:  Procedure ClearRecentDocs
6554:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6555:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6556:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6557:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6558:  Function RecycleFile( FileToRecycle : string) : Boolean
6559:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6560:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6561:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6562:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6563:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6564:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6565:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6566:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6567:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6568:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6569:
6570: ****************************************** unit uPSI_JclPeImage;
6571:
6572:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6573: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6574:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6575:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6576:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6577:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6578:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6579:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6580:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6581:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Bool
6582:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6583:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6584:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6585:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
      LibraryName:string; IncludeLibNames : Boolean): Boolean
6586:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
```

```
6587:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6588:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6589:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
       NamesList:TStrings):Bool
6590:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6591:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
       Descript:Bool):Bool;
6592:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6593:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6594:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6595:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6596:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6597:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6598:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
       PImageSectionHeader
6599:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6600:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6601:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
       ___Pointer;
6602:    SIRegister_TJclPeSectionStream(CL);
6603:    SIRegister_TJclPeMapImgHookItem(CL);
6604:    SIRegister_TJclPeMapImgHooks(CL);
6605:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6606:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6607:    Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6608:    TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6609:    TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6610:    TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6611:    TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6612:    TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6613:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6614:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6615:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6616:  Function PeBorUnmangleName3( const Name : string) : string;
6617:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6618:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6619:
6620:
6621: //****************** SysTools uPSI_StSystem; ************************************
6622:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6623:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6624:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6625:  //Procedure EnumerateDirectories(const
       StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6626:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6627:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6628:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6629:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6630:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6631:  Function FlushOsBuffers( Handle : Integer) : Boolean
6632:  Function GetCurrentUser : AnsiString
6633:  Function GetDiskClass( Drive : Char) : DiskClass
6634:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6635:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6636:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6637:    { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6638:  Function getDiskSpace2(const path: String; index: integer): int64;
6639:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6640:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6641:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6642:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6643:  Function GetLongPath( const APath : AnsiString) : AnsiString
6644:  Function GetMachineName : AnsiString
6645:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6646:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6647:  Function GetShortPath( const APath : AnsiString) : AnsiString
6648:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6649:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6650:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6651:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6652:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6653:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6654:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6655:  Function IsDriveReady( Drive : Char) : Boolean
6656:  Function IsFile( const FileName : AnsiString) : Boolean
6657:  Function IsFileArchive( const S : AnsiString) : Integer
6658:  Function IsFileHidden( const S : AnsiString) : Integer
6659:  Function IsFileReadOnly( const S : AnsiString) : Integer
6660:  Function IsFileSystem( const S : AnsiString) : Integer
6661:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6662:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6663:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
```

```
6664:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6665:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6666:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6667:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6668:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6669:  Function ValidDrive( Drive : Char) : Boolean
6670:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6671:
6672: //*****************************unit uPSI_JclLANMan;*****************************************
6673: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6674:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6675:  Function DeleteAccount( const Servername, Username : string) : Boolean
6676:  Function DeleteLocalAccount( Username : string) : Boolean
6677:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6678:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6679:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6680:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6681:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6682:  Function LocalGroupExists( const Group : string) : Boolean
6683:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6684:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6685:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6686:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6687:  Function IsLocalAccount( const AccountName : string) : Boolean
6688:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6689:  Function GetRandomString( NumChar : cardinal) : string
6690:
6691: //*****************************unit uPSI_cUtils;*****************************************
6692: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6693: Function cIsWinNT : boolean
6694:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6695:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6696:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6697:  Function cGetShortName( FileName : string) : string
6698:  Procedure cShowError( Msg : String)
6699:  Function cCommaStrToStr( s : string; formatstr : string) : string
6700:  Function cIncludeQuoteIfSpaces( s : string) : string
6701:  Function cIncludeQuoteIfNeeded( s : string) : string
6702:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6703:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6704:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6705:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6706:  Function cCodeInstoStr( s : string) : string
6707:  Function cStrtoCodeIns( s : string) : string
6708:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6709:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6710:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6711:  Function cPointtoStr( const pt : TPoint) : string
6712:  Function cListtoStr( const List : TStrings) : string
6713:  Function ListtoStr( const List : TStrings) : string
6714:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6715:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6716:  Function cGetFileTyp( const FileName : string) : TUnitType
6717:  Function cGetExTyp( const FileName : string) : TExUnitType
6718:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6719:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6720:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6721:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6722:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6723:  Function cGenMakePath( FileName : String) : String;
6724:  Function cGenMakePath2( FileName : String) : String
6725:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6726:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6727:  Function cCalcMod( Count : Integer) : Integer
6728:  Function cGetVersionString( FileName : string) : string
6729:  Function cCheckChangeDir( var Dir : string) : boolean
6730:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6731:  Function cIsNumeric( s : string) : boolean
6732:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6733:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6734:  Function GetFileTyp( const FileName : string) : TUnitType
6735:  Function Atoi(const aStr: string): integer
6736:  Function Itoa(const aint: integer): string
6737:  Function Atof(const aStr: string): double');
6738:  Function Atol(const aStr: string): longint');
6739:
6740:
6741: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6742: begin
6743:   FindClass('TOBJECT'),'EHTTP
6744:   FindClass('TOBJECT'),'EHTTPParser
6745:   //AnsiCharSet', 'set of AnsiChar
6746:   AnsiStringArray', 'array of AnsiString
6747:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6748:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
```

```
6749:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6750:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6751:    +'CustomMinVersion : Integer; end
6752:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6753:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6754:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6755:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6756:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6757:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6758:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6759:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6760:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6761:    +'nection, hntOrigin, hntKeepAlive )
6762:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6763:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6764:    +' AnsiString; end
6765:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6766:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6767:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6768:   //PHTTPContentLength', '^THTTPContentLength // will not work
6769:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6770:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6771:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6772:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6773:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6774:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6775:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6776:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6777:    +' CustomStr : AnsiString; end
6778:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6779:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6780:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6781:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6782:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6783:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6784:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6785:    +'m; Custom : AnsiString; end
6786:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6787:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6788:    +' Custom : AnsiString; end
6789:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6790:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6791:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6792:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6793:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6794:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6795:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6796:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6797:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6798:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6799:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6800:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6801:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6802:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6803:    +'FieldEnum; List : array of THTTPContentEncoding; end
6804:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6805:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6806:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6807:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6808:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6809:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6810:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6811:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6812:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6813:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6814:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6815:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6816:    +'CustomFieldArray; Custom : AnsiString; end
6817:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6818:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6819:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6820:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6821:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6822:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6823:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6824:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6825:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6826:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6827:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6828:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6829:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6830:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6831:   THTTPFixedHeaders','array[0..42] of AnsiString
6832:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6833:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6834:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6835:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6836:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6837:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
```

```
6838:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6839:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6840:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6841:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6842:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6843:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6844:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6845:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6846:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6847:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6848:    +' THTTPDateField; Age : THTTPAgeField; end
6849:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6850:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6851:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6852:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6853:  Procedure InitHTTPRequest( var A : THTTPRequest)
6854:  Procedure InitHTTPResponse( var A : THTTPResponse)
6855:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6856:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6857:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6858:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6859:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6860:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6861:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6862:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6863:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6864:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6865:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6866:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6867:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6868:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6869:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6870:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6871:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6872:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6873:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6874:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6875:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6876:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6877:   THTTPStringOption', '( hsoNone )
6878:   THTTPStringOptions', 'set of THTTPStringOption
6879:   FindClass('TOBJECT'),'TAnsiStringBuilder
6880:
6881:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6882:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6883:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6884:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6885:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6886:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6887:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6888:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6889:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6890:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6891:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6892:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6893:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6894:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6895:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6896:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6897:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6898:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6899:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6900:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6901:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6902:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6903:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6904:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
```

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

```
6985:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6986:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6987:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6988:    +'nteger; Supported : Boolean; end
6989:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6990:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6991:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6992:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6993:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6994:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6995:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6996:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6997:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6998:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6999:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
7000:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
7001:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
7002:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
7003:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
7004:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
7005:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
7006:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
7007:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
7008:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
7009:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
7010:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
7011:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
7012:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
7013:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
7014:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7015:  Procedure SelfTestcTLSUtils
7016: end;
7017:
7018: (*----------------------------------------------------------------------------*)
7019: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
7020: begin
7021:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
7022:  // pBoard', '^tBoard // will not work
7023:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
7024:  Function rCheckMove( color : byte; cx, cy : integer) : integer
7025:  //Function rDoStep( data : pBoard) : word
7026:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
7027: end;
7028:
7029: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7030: begin
7031: Function InEditMode( ADataset : TDataset) : Boolean
7032: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
7033: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7034: Function GetFieldText( AField : TField) : String
7035: end;
7036:
7037: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
7038: begin
7039:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7040:   TMyPrintRange', '( prAll, prSelected )
7041:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7042:    +'ded, ssDateTime, ssTime, ssCustom )
7043:   TSortDirection', '( sdAscending, sdDescending )
7044:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7045:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7046:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7047:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7048:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7049:   SIRegister_TSortOptions(CL);
7050:   SIRegister_TPrintOptions(CL);
7051:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7052:   SIRegister_TSortedList(CL);
7053:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7054:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7055:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7056:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7057:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7058:   SIRegister_TFontSetting(CL);
7059:   SIRegister_TFontList(CL);
7060:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7061:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7062:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7063:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7064:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
```

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

```
7153:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7154:  Function XRTLSwapHiLo16( X : Word) : Word
7155:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7156:  Function XRTLSwapHiLo64( X : Int64) : Int64
7157:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7158:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7159:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7160:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7161:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7162:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7163:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7164:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7165:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7166:  Function XRTLPopulation( A : Cardinal) : Cardinal
7167: end;
7168:
7169: Function XRTLURLDecode( const ASrc : WideString) : WideString
7170: Function XRTLURLEncode( const ASrc : WideString) : WideString
7171: Function XRTLURINormalize( const AURI : WideString) : WideString
7172: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7173: Function XRTLExtractLongPathName(APath: string): string;
7174:
7175: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7176: begin
7177:   AddTypeS('Int8', 'ShortInt
7178:   AddTypeS('Int16', 'SmallInt
7179:   AddTypeS('Int32', 'LongInt
7180:   AddTypeS('UInt8', 'Byte
7181:   AddTypeS('UInt16', 'Word
7182:   AddTypeS('UInt32', 'LongWord
7183:   AddTypeS('UInt64', 'Int64
7184:   AddTypeS('Word8', 'UInt8
7185:   AddTypeS('Word16', 'UInt16
7186:   AddTypeS('Word32', 'UInt32
7187:   AddTypeS('Word64', 'UInt64
7188:   AddTypeS('LargeInt', 'Int64
7189:   AddTypeS('NativeInt', 'Integer
7190:   AddTypeS('NativeUInt', 'Cardinal
7191:   Const('BitsPerByte','LongInt'( 8);
7192:   Const('BitsPerWord','LongInt'( 16);
7193:   Const('BitsPerLongWord','LongInt'( 32);
7194:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7195:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7196:  Function MinI( const A, B : Integer) : Integer
7197:  Function MaxI( const A, B : Integer) : Integer
7198:  Function MinC( const A, B : Cardinal) : Cardinal
7199:  Function MaxC( const A, B : Cardinal) : Cardinal
7200:  Function SumClipI( const A, I : Integer) : Integer
7201:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7202:  Function InByteRange( const A : Int64) : Boolean
7203:  Function InWordRange( const A : Int64) : Boolean
7204:  Function InLongWordRange( const A : Int64) : Boolean
7205:  Function InShortIntRange( const A : Int64) : Boolean
7206:  Function InSmallIntRange( const A : Int64) : Boolean
7207:  Function InLongIntRange( const A : Int64) : Boolean
7208:   AddTypeS('Bool8', 'ByteBool
7209:   AddTypeS('Bool16', 'WordBool
7210:   AddTypeS('Bool32', 'LongBool
7211:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7212:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7213:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7214:  Const('MinSingle','Single').setExtended( 1.5E-45);
7215:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7216:  Const('MinDouble','Double').setExtended( 5.0E-324);
7217:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7218:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7219:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7220:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7221:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7222:  Function MinF( const A, B : Float) : Float
7223:  Function MaxF( const A, B : Float) : Float
7224:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7225:  Function InSingleRange( const A : Float) : Boolean
7226:  Function InDoubleRange( const A : Float) : Boolean
7227:  Function InCurrencyRange( const A : Float) : Boolean;
7228:  Function InCurrencyRange1( const A : Int64) : Boolean;
7229:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7230:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7231:  Function FloatIsInfinity( const A : Extended) : Boolean
7232:  Function FloatIsNaN( const A : Extended) : Boolean
7233:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7234:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7235:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7236:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7237:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7238:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7239:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7240:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
```

```
7241:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7242:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7243:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7244:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7245:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7246:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7247:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7248:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7249:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7250:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7251:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7252:  Function SetBitScanForward( const Value : LongWord) : Integer;
7253:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7254:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7255:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7256:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7257:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7258:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7259:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7260:  Function cReverseBits( const Value : LongWord) : LongWord;
7261:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7262:  Function cSwapEndian( const Value : LongWord) : LongWord
7263:  Function cTwosComplement( const Value : LongWord) : LongWord
7264:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7265:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7266:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7267:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7268:  Function cBitCount( const Value : LongWord) : LongWord
7269:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7270:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7271:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7272:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7273:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7274:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7275:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7276:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7277:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7278: //  AddTypeS('CharSet', 'set of AnsiChar
7279:   AddTypeS('CharSet', 'set of Char       //!!!
7280:   AddTypeS('AnsiCharSet', 'TCharSet
7281:   AddTypeS('ByteSet', 'set of Byte
7282:   AddTypeS('AnsiChar', 'Char
7283:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7284:  Function AsByteSet( const C : array of Byte) : ByteSet
7285:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7286:  Procedure ClearCharSet( var C : CharSet)
7287:  Procedure FillCharSet( var C : CharSet)
7288:  procedure FillCharSearchRec;  // with 0
7289:  Procedure ComplementCharSet( var C : CharSet)
7290:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7291:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7292:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7293:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7294:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7295:  Function IsSubSet( const A, B : CharSet) : Boolean
7296:  Function IsEqual( const A, B : CharSet) : Boolean
7297:  Function IsEmpty( const C : CharSet) : Boolean
7298:  Function IsComplete( const C : CharSet) : Boolean
7299:  Function cCharCount( const C : CharSet) : Integer
7300:  Procedure ConvertCaseInsensitive( var C : CharSet)
7301:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7302:  Function IntRangeLength( const Low, High : Integer) : Int64
7303:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7304:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7305:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7306:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7307:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7308:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7309:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7310:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7311:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7312:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7313:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7314:   AddTypeS('UnicodeChar', 'WideChar
7315:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7316:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7317:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7318:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7319:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7320:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7321:  Function cSgn( const A : LongInt) : Integer;
7322:  Function cSgn1( const A : Int64) : Integer;
7323:  Function cSgn2( const A : Extended) : Integer;
7324:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7325:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7326:  Function WideCharToInt( const A : WideChar) : Integer
7327:  Function CharToInt( const A : Char) : Integer
7328:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7329:  Function IntToWideChar( const A : Integer) : WideChar
```

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

```
7419:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7420:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7421:  unit uPSI_cFundamentUtils;
7422:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7423:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7424:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7425:  Const('CCHARSET','Stringb64_XXEncode);
7426:  Const('CHEXSET','String'0123456789ABCDEF
7427:  Const('HEXDIGITS','String'0123456789ABCDEF
7428:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7429:  Const('DIGISET','String'0123456789
7430:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7431:  Const('DIGISET2','TCharset').SetSet('0123456789'
7432:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7433:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7434:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7435:  Const('NUMBERS','String'0123456789');
7436:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7437:  Function CharSetToStr( const C : CharSet) : AnsiString
7438:  Function StrToCharSet( const S : AnsiString) : CharSet
7439:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7440:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7441:  Function UUDecode( const S : AnsiString) : AnsiString
7442:  Function XXDecode( const S : AnsiString) : AnsiString
7443:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7444:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7445:  Function InterfaceToStrW( const I : IInterface) : WideString
7446:  Function InterfaceToStr( const I : IInterface) : String
7447:  Function ObjectClassName( const O : TObject) : String
7448:  Function ClassClassName( const C : TClass) : String
7449:  Function ObjectToStr( const O : TObject) : String
7450:  Function ObjectToString( const O : TObject) : String
7451:  Function CharSetToStr( const C : CharSet) : AnsiString
7452:  Function StrToCharSet( const S : AnsiString) : CharSet
7453:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7454:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7455:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7456:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7457:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7458:  Const('Bytes1KB','LongInt'( 1024);
7459:   SIRegister_IInterface(CL);
7460:  Procedure SelfTestCFundamentUtils
7461:
7462: Function CreateSchedule : IJclSchedule
7463: Function NullStamp : TTimeStamp
7464: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7465: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7466: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7467:
7468: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7469: begin
7470:  AddTypeS('TFunc', 'function(X : Float) : Float;
7471: Function InitGraphics( Width, Height : Integer) : Boolean
7472: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7473: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7474: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7475: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7476: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7477: Procedure SetGraphTitle( Title : String)
7478: Procedure SetOxTitle( Title : String)
7479: Procedure SetOyTitle( Title : String)
7480: Function GetGraphTitle : String
7481: Function GetOxTitle : String
7482: Function GetOyTitle : String
7483: Procedure PlotOxAxis( Canvas : TCanvas)
7484: Procedure PlotOyAxis( Canvas : TCanvas)
7485: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7486: Procedure WriteGraphTitle( Canvas : TCanvas)
7487: Function SetMaxCurv( NCurv : Byte) : Boolean
7488: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7489: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7490: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7491: Procedure SetCurvStep( CurvIndex, Step : Integer)
7492: Function GetMaxCurv : Byte
7493: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7494: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7495: Function GetCurvLegend( CurvIndex : Integer) : String
7496: Function GetCurvStep( CurvIndex : Integer) : Integer
7497: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7498: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7499: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7500: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7501: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7502: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7503: Function Xpixel( X : Float) : Integer
```

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

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

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

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

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

```
7941:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7942:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7943:  Procedure ShowMessageUser( const S : String)
7944:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7945:  Function HasLabels( ASeries : TChartSeries) : Boolean
7946:  Function HasColors( ASeries : TChartSeries) : Boolean
7947:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7948:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7949: end;
7950:
7951:
7952: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7953: begin
7954:  //'TeeFormBorderStyle',' bsNone);
7955:   SIRegister_TMetafile(CL);
7956:  'TeeDefVerticalMargin','LongInt'( 4);
7957:  'TeeDefHorizMargin','LongInt'( 3);
7958:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7959:  'TeeMsg_TeeHand','String 'crTeeHand
7960:  'TeeNormalPrintDetail','LongInt'( 0);
7961:  'TeeHighPrintDetail','LongInt'( - 100);
7962:  'TeeDefault_PrintMargin','LongInt'( 15);
7963:  'MaxDefaultColors','LongInt'( 19);
7964:  'TeeTabDelimiter','Char #9);
7965:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7966:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7967:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7968:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7969:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7970:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7971:   SIRegister_TCustomPanelNoCaption(CL);
7972:   FindClass('TOBJECT'),'TCustomTeePanel
7973:   SIRegister_TZoomPanning(CL);
7974:   SIRegister_TTeeEvent(CL);
7975:   //SIRegister_TTeeEventListeners(CL);
7976:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7977:   SIRegister_TTeeMouseEvent(CL);
7978:   SIRegister_TCustomTeePanel(CL);
7979:   //TChartGradient', 'TTeeGradient
7980:   //TChartGradientClass', 'class of TChartGradient
7981:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7982:   SIRegister_TTeeZoomPen(CL);
7983:   SIRegister_TTeeZoomBrush(CL);
7984:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7985:   SIRegister_TTeeZoom(CL);
7986:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7987:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7988:   SIRegister_TBackImage(CL);
7989:   SIRegister_TCustomTeePanelExtended(CL);
7990:   //TChartBrushClass', 'class of TChartBrush
7991:   SIRegister_TTeeCustomShapeBrushPen(CL);
7992:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7993:   TTextFormat', '( ttfNormal, ttfHtml )
7994:   SIRegister_TTeeCustomShape(CL);
7995:   SIRegister_TTeeShape(CL);
7996:   SIRegister_TTeeExportData(CL);
7997:  Function TeeStr( const Num : Integer) : String
7998:  Function DateTimeDefaultFormat( const AStep : Double) : String
7999:  Function TEEDaysInMonth( Year, Month : Word) : Word
8000:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
8001:  Function NextDateTimeStep( const AStep : Double) : Double
8002:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
8003:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
8004:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
8005:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
8006:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
8007:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
8008:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
8009:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
8010:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
8011:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
8012:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
8013:  Function DelphiToLocalFormat( const Format : String) : String
8014:  Function LocalToDelphiFormat( const Format : String) : String
8015:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8016:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8017:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8018:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
8019:   TTeeSortSwap', 'Procedure ( a, b : Integer)
8020:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8021:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
8022:  Function TeeExtractField( St : String; Index : Integer) : String;
8023:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
8024:  Function TeeNumFields( St : String) : Integer;
8025:  Function TeeNumFields1( const St, Separator : String) : Integer;
8026:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8027:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8028:  // TColorArray', 'array of TColor
```

```
8029:  Function GetDefaultColor( const Index : Integer) : TColor
8030:  Procedure SetDefaultColorPalette;
8031:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
8032:  'TeeCheckBoxSize','LongInt'( 11);
8033:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8034:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
8035:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8036:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8037:  Procedure TeeTranslateControl( AControl : TControl);
8038:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
8039:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8040:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8041:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8042:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8043:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8044:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8045:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8046:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8047:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8048:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8049:  Procedure TeeSaveStringOption( const AKey, Value : String)
8050:  Function TeeDefaultXMLEncoding : String
8051:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8052:   TeeWindowHandle', 'Integer
8053:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8054:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8055:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8056: end;
8057:
8058:
8059: using mXBDEUtils
8060: ********************************************************************************
8061:  Procedure SetAlias( aAlias, aDirectory : String)
8062:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
8063:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8064:  Procedure SetBDE( aPath, aNode, aValue : String)
8065:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8066:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8067:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8068:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8069:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8070:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8071:
8072:
8073: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8074: begin
8075: AddClassN(FindClass('TOBJECT'),'EDateTime
8076: Function DatePart( const D : TDateTime) : Integer
8077: Function TimePart( const D : TDateTime) : Double
8078: Function Century( const D : TDateTime) : Word
8079: Function Year( const D : TDateTime) : Word
8080: Function Month( const D : TDateTime) : Word
8081: Function Day( const D : TDateTime) : Word
8082: Function Hour( const D : TDateTime) : Word
8083: Function Minute( const D : TDateTime) : Word
8084: Function Second( const D : TDateTime) : Word
8085: Function Millisecond( const D : TDateTime) : Word
8086: ('OneDay','Extended').setExtended( 1.0);
8087: ('OneHour','Extended').SetExtended( OneDay / 24);
8088: ('OneMinute','Extended').SetExtended( OneHour / 60);
8089: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8090: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8091: ('OneWeek','Extended').SetExtended( OneDay * 7);
8092: ('HoursPerDay','Extended').SetExtended( 24);
8093: ('MinutesPerHour','Extended').SetExtended( 60);
8094: ('SecondsPerMinute','Extended').SetExtended( 60);
8095: Procedure SetYear( var D : TDateTime; const Year : Word)
8096: Procedure SetMonth( var D : TDateTime; const Month : Word)
8097: Procedure SetDay( var D : TDateTime; const Day : Word)
8098: Procedure SetHour( var D : TDateTime; const Hour : Word)
8099: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8100: Procedure SetSecond( var D : TDateTime; const Second : Word)
8101: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8102: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8103: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8104: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8105: Function IsAM( const D : TDateTime) : Boolean
8106: Function IsPM( const D : TDateTime) : Boolean
8107: Function IsMidnight( const D : TDateTime) : Boolean
8108: Function IsNoon( const D : TDateTime) : Boolean
8109: Function IsSunday( const D : TDateTime) : Boolean
8110: Function IsMonday( const D : TDateTime) : Boolean
8111: Function IsTuesday( const D : TDateTime) : Boolean
8112: Function IsWedneday( const D : TDateTime) : Boolean
8113: Function IsThursday( const D : TDateTime) : Boolean
8114: Function IsFriday( const D : TDateTime) : Boolean
8115: Function IsSaturday( const D : TDateTime) : Boolean
8116: Function IsWeekend( const D : TDateTime) : Boolean
```

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

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

```
8294:
8295: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8296: begin
8297:  Function RandomSeed : LongWord
8298:  Procedure AddEntropy( const Value : LongWord)
8299:  Function RandomUniform : LongWord;
8300:  Function RandomUniform1( const N : Integer) : Integer;
8301:  Function RandomBoolean : Boolean
8302:  Function RandomByte : Byte
8303:  Function RandomByteNonZero : Byte
8304:  Function RandomWord : Word
8305:  Function RandomInt64 : Int64;
8306:  Function RandomInt641( const N : Int64) : Int64;
8307:  Function RandomHex( const Digits : Integer) : String
8308:  Function RandomFloat : Extended
8309:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8310:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8311:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8312:  Function mwcRandomLongWord : LongWord
8313:  Function urnRandomLongWord : LongWord
8314:  Function moaRandomFloat : Extended
8315:  Function mwcRandomFloat : Extended
8316:  Function RandomNormalF : Extended
8317:  Procedure SelfTestCRandom
8318: end;
8319:
8320: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8321: begin
8322:  // PIntArray', '^TIntArray // will not work
8323:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8324:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8325:  Function synMax( x, y : integer) : integer
8326:  Function synMin( x, y : integer) : integer
8327:  Function synMinMax( x, mi, ma : integer) : integer
8328:  Procedure synSwapInt( var l, r : integer)
8329:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8330:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8331:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8332:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8333:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8334:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8335:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8336:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8337:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8338:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8339:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8340:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8341:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8342:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8343:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8344:  ('C3_NONSPACING','LongInt'( 1);
8345:  'C3_DIACRITIC','LongInt'( 2);
8346:  'C3_VOWELMARK','LongInt'( 4);
8347:  ('C3_SYMBOL','LongInt'( 8);
8348:  ('C3_KATAKANA','LongWord( $0010);
8349:  ('C3_HIRAGANA','LongWord( $0020);
8350:  ('C3_HALFWIDTH','LongWord( $0040);
8351:  ('C3_FULLWIDTH','LongWord( $0080);
8352:  ('C3_IDEOGRAPH','LongWord( $0100);
8353:  ('C3_KASHIDA','LongWord( $0200);
8354:  ('C3_LEXICAL','LongWord( $0400);
8355:  ('C3_ALPHA','LongWord( $8000);
8356:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8357:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8358:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8359:  Function synIsStringType( Value : Word) : TStringType
8360:  Function synGetEOL( Line : PChar) : PChar
8361:  Function synEncodeString( s : string) : string
8362:  Function synDecodeString( s : string) : string
8363:  Procedure synFreeAndNil( var Obj: TObject)
8364:  Procedure synAssert( Expr : Boolean)
8365:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8366:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8367:   TReplaceFlags', 'set of TReplaceFlag )
8368:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8369:  Function synGetRValue( RGBValue : TColor) : byte
8370:  Function synGetGValue( RGBValue : TColor) : byte
8371:  Function synGetBValue( RGBValue : TColor) : byte
8372:  Function synRGB( r, g, b : Byte) : Cardinal
8373: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8374:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8375:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8376:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8377:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8378: end;
8379:
8380:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
```

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

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

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

```
8644:  Function RedComponent( Color32 : TColor32) : Integer
8645:  Function GreenComponent( Color32 : TColor32) : Integer
8646:  Function BlueComponent( Color32 : TColor32) : Integer
8647:  Function AlphaComponent( Color32 : TColor32) : Integer
8648:  Function Intensity( Color32 : TColor32) : Integer
8649:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8650:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8651:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8652:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8653:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8654:  Function WinPalette( const P : TPalette32) : HPALETTE
8655:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8656:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8657:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8658:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8659:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8660:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8661:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8662:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8663:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8664:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8665:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8666:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8667:  Function FixedRect1( const ARect : TRect) : TRect;
8668:  Function FixedRect2( const FR : TFloatRect) : TRect;
8669:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8670:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8671:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8672:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8673:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8674:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8675:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8676:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8677:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8678:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8679:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8680:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8681:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8682:  Function IsRectEmpty( const R : TRect) : Boolean;
8683:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8684:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8685:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8686:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8687:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8688:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8689:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8690:  Function MessageBeep( uType : UINT) : BOOL
8691:  Function ShowCursor( bShow : BOOL) : Integer
8692:  Function SetCursorPos( X, Y : Integer) : BOOL
8693:  Function SetCursor( hCursor : HICON) : HCURSOR
8694:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8695:  //Function ClipCursor( lpRect : PRect) : BOOL
8696:  Function GetClipCursor( var lpRect : TRect) : BOOL
8697:  Function GetCursor : HCURSOR
8698:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8699:  Function GetCaretBlinkTime : UINT
8700:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8701:  Function DestroyCaret : BOOL
8702:  Function HideCaret( hWnd : HWND) : BOOL
8703:  Function ShowCaret( hWnd : HWND) : BOOL
8704:  Function SetCaretPos( X, Y : Integer) : BOOL
8705:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8706:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8707:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8708:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8709:  Function WindowFromPoint( Point : TPoint) : HWND
8710:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8711:
8712:
8713: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8714: begin
8715:  Function FixedFloor( A : TFixed) : Integer
8716:  Function FixedCeil( A : TFixed) : Integer
8717:  Function FixedMul( A, B : TFixed) : TFixed
8718:  Function FixedDiv( A, B : TFixed) : TFixed
8719:  Function OneOver( Value : TFixed) : TFixed
8720:  Function FixedRound( A : TFixed) : Integer
8721:  Function FixedSqr( Value : TFixed) : TFixed
8722:  Function FixedSqrtLP( Value : TFixed) : TFixed
8723:  Function FixedSqrtHP( Value : TFixed) : TFixed
8724:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8725:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8726:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8727:  Function GRHypot( const X, Y : TFloat) : TFloat;
8728:  Function Hypot1( const X, Y : Integer) : Integer;
8729:  Function FastSqrt( const Value : TFloat) : TFloat
8730:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8731:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8732:  Function FastInvSqrt( const Value : Single) : Single;
```

```
8733:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8734:  Function GRIsPowerOf2( Value : Integer) : Boolean
8735:  Function PrevPowerOf2( Value : Integer) : Integer
8736:  Function NextPowerOf2( Value : Integer) : Integer
8737:  Function Average( A, B : Integer) : Integer
8738:  Function GRSign( Value : Integer) : Integer
8739:  Function FloatMod( x, y : Double) : Double
8740: end;
8741:
8742: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8743: begin
8744:  Function Clamp( const Value : Integer) : Integer;
8745:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8746:  Function StackAlloc( Size : Integer) : Pointer
8747:  Procedure StackFree( P : Pointer)
8748:  Procedure Swap( var A, B : Pointer);
8749:  Procedure Swap1( var A, B : Integer);
8750:  Procedure Swap2( var A, B : TFixed);
8751:  Procedure Swap3( var A, B : TColor32);
8752:  Procedure TestSwap( var A, B : Integer);
8753:  Procedure TestSwap1( var A, B : TFixed);
8754:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8755:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8756:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8757:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8758:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8759:  Function GRMin( const A, B, C : Integer) : Integer;
8760:  Function GRMax( const A, B, C : Integer) : Integer;
8761:  Function Clamp( Value, Max : Integer) : Integer;
8762:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8763:  Function Wrap( Value, Max : Integer) : Integer;
8764:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8765:  Function Wrap3( Value, Max : Single) : Single;;
8766:  Function WrapPow2( Value, Max : Integer) : Integer;
8767:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8768:  Function Mirror( Value, Max : Integer) : Integer;
8769:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8770:  Function MirrorPow2( Value, Max : Integer) : Integer;
8771:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8772:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8773:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8774:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8775:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8776:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8777:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8778:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8779:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8780:  Function Div255( Value : Cardinal) : Cardinal
8781:  Function SAR_4( Value : Integer) : Integer
8782:  Function SAR_8( Value : Integer) : Integer
8783:  Function SAR_9( Value : Integer) : Integer
8784:  Function SAR_11( Value : Integer) : Integer
8785:  Function SAR_12( Value : Integer) : Integer
8786:  Function SAR_13( Value : Integer) : Integer
8787:  Function SAR_14( Value : Integer) : Integer
8788:  Function SAR_15( Value : Integer) : Integer
8789:  Function SAR_16( Value : Integer) : Integer
8790:  Function ColorSwap( WinColor : TColor) : TColor32
8791: end;
8792:
8793: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8794: begin
8795:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8796:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8797:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8798:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8799:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8800:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8801:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8802:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8803:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8804:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8805:  Function CreateBitmask( Components : TColor32Components) : TColor32
8806:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8807:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8808:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8809: end;
8810:
8811:
8812: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8813: begin
8814:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8815:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8816:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8817:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8818:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
```

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

```
8905:   SIRegister_TJclLinearTransformation(CL);
8906:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8907:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8908:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8909:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8910:  Procedure BitmapToJPeg( const FileName : string)
8911:  Procedure JPegToBitmap( const FileName : string)
8912:  Function ExtractIconCount( const FileName : string) : Integer
8913:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8914:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8915:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8916:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8917:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8918:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8919:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8920:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8921:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8922:  Procedure ScreenShot1( bm : TBitmap);
8923:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8924:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8925:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8926:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8927:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8928:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8929:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8930:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8931:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8932:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8933:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8934:  Procedure Invert( Dst, Src : TJclBitmap32)
8935:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8936:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8937:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8938:  Procedure SetGamma( Gamma : Single)
8939: end;
8940:
8941: (*----------------------------------------------------------------------------*)
8942: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8943: begin
8944:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8945:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8946:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8947:  Function LockedDec( var Target : Integer) : Integer
8948:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8949:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8950:  Function LockedExchangeDec( var Target : Integer) : Integer
8951:  Function LockedExchangeInc( var Target : Integer) : Integer
8952:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8953:  Function LockedInc( var Target : Integer) : Integer
8954:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8955:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8956:   SIRegister_TJclDispatcherObject(CL);
8957:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8958:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8959:   SIRegister_TJclCriticalSection(CL);
8960:   SIRegister_TJclCriticalSectionEx(CL);
8961:   SIRegister_TJclEvent(CL);
8962:   SIRegister_TJclWaitableTimer(CL);
8963:   SIRegister_TJclSemaphore(CL);
8964:   SIRegister_TJclMutex(CL);
8965:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8966:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8967:   SIRegister_TJclOptex(CL);
8968:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8969:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8970:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8971:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8972:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8973:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8974:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8975:   PMeteredSection', '^TMeteredSection // will not work
8976:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8977:   SIRegister_TJclMeteredSection(CL);
8978:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8979:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8980:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8981:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8982:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8983:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8984:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8985:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8986:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
```

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

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

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

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

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

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

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

```
9610: { 53-bit random real in [0,1) }
9611: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9612: { Initializes the 'Multiply with carry' random number generator }
9613: function IRanMWC : RNG_IntType; external 'dmath';
9614: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9615: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9616: { Initializes Mersenne Twister generator with a seed }
9617: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9618:                         KeyLength : Word); external 'dmath';
9619: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9620: function IRanMT : RNG_IntType; external 'dmath';
9621: { Random integer from MT generator }
9622: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9623: { Initializes the UVAG generator with a string }
9624: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9625: { Initializes the UVAG generator with an integer }
9626: function IRanUVAG : RNG_IntType; external 'dmath';
9627: { Random integer from UVAG generator }
9628: function RanGaussStd : Float; external 'dmath';
9629: { Random number from standard normal distribution }
9630: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9631: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9632: procedure RanMult(M      : TVector; L      : TMatrix;
9633:                   Lb, Ub : Integer;
9634:                   X      : TVector); external 'dmath';
9635: { Random vector from multinormal distribution (correlated) }
9636: procedure RanMultIndep(M, S   : TVector;
9637:                        Lb, Ub : Integer;
9638:                        X      : TVector); external 'dmath';
9639: { Random vector from multinormal distribution (uncorrelated) }
9640: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9641: { Initializes Metropolis-Hastings parameters }
9642: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9643: { Returns Metropolis-Hastings parameters }
9644: procedure Hastings(Func      : TFuncNVar;
9645:                    T         : Float;
9646:                    X         : TVector;
9647:                    V         : TMatrix;
9648:                    Lb, Ub    : Integer;
9649:                    Xmat      : TMatrix;
9650:                    X_min     : TVector;
9651:                    var F_min : Float); external 'dmath';
9652: { Simulation of a probability density function by Metropolis-Hastings }
9653: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9654: { Initializes Simulated Annealing parameters }
9655: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9656: { Initializes log file }
9657: procedure SimAnn(Func          : TFuncNVar;
9658:                  X, Xmin, Xmax : TVector;
9659:                  Lb, Ub        : Integer;
9660:                  var F_min     : Float); external 'dmath';
9661: { Minimization of a function of several var. by simulated annealing }
9662: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9663: { Initializes Genetic Algorithm parameters }
9664: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9665: { Initializes log file }
9666: procedure GenAlg(Func          : TFuncNVar;
9667:                  X, Xmin, Xmax : TVector;
9668:                  Lb, Ub        : Integer;
9669:                  var F_min     : Float); external 'dmath';
9670: { Minimization of a function of several var. by genetic algorithm }
9671: { ------------------------------------------------------------------
9672:   Statistics
9673:   ------------------------------------------------------------------ }
9674: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9675: { Mean of sample X }
9676: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9677: { Minimum of sample X }
9678: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9679: { Maximum of sample X }
9680: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9681: { Median of sample X }
9682: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9683: { Standard deviation estimated from sample X }
9684: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9685: { Standard deviation of population }
9686: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9687: { Correlation coefficient }
9688: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9689: { Skewness of sample X }
9690: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9691: { Kurtosis of sample X }
9692: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9693: { Quick sort (ascending order) }
9694: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9695: { Quick sort (descending order) }
9696: procedure Interval(X1, X2             : Float;
9697:                    MinDiv, MaxDiv     : Integer;
9698:                    var Min, Max, Step : Float); external 'dmath';
```

```
9699: { Determines an interval for a set of values }
9700: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9701:                     var XMin, XMax, XStep : Float); external 'dmath';
9702: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9703: procedure StudIndep(N1, N2         : Integer;
9704:                     M1, M2, S1, S2 : Float;
9705:                     var T          : Float;
9706:                     var DoF        : Integer); external 'dmath';
9707: { Student t-test for independent samples }
9708: procedure StudPaired(X, Y    : TVector;
9709:                      Lb, Ub  : Integer;
9710:                      var T   : Float;
9711:                      var DoF : Integer); external 'dmath';
9712: { Student t-test for paired samples }
9713: procedure AnOVa1(Ns               : Integer;
9714:                  N                : TIntVector;
9715:                  M, S             : TVector;
9716:                  var V_f, V_r, F  : Float;
9717:                  var DoF_f, DoF_r : Integer); external 'dmath';
9718: { One-way analysis of variance }
9719: procedure AnOVa2(NA, NB, Nobs : Integer;
9720:                  M, S         : TMatrix;
9721:                  V, F         : TVector;
9722:                  DoF          : TIntVector); external 'dmath';
9723: { Two-way analysis of variance }
9724: procedure Snedecor(N1, N2         : Integer;
9725:                    S1, S2         : Float;
9726:                    var F          : Float;
9727:                    var DoF1, DoF2 : Integer); external 'dmath';
9728: { Snedecor's F-test (comparison of two variances) }
9729: procedure Bartlett(Ns       : Integer;
9730:                    N        : TIntVector;
9731:                    S        : TVector;
9732:                    var Khi2 : Float;
9733:                    var DoF  : Integer); external 'dmath';
9734: { Bartlett's test (comparison of several variances) }
9735: procedure Mann_Whitney(N1, N2     : Integer;
9736:                        X1, X2     : TVector;
9737:                        var U, Eps : Float); external 'dmath';
9738: { Mann-Whitney test}
9739: procedure Wilcoxon(X, Y       : TVector;
9740:                    Lb, Ub     : Integer;
9741:                    var Ndiff  : Integer;
9742:                    var T, Eps : Float); external 'dmath';
9743: { Wilcoxon test }
9744: procedure Kruskal_Wallis(Ns      : Integer;
9745:                          N       : TIntVector;
9746:                          X       : TMatrix;
9747:                          var H   : Float;
9748:                          var DoF : Integer); external 'dmath';
9749: { Kruskal-Wallis test }
9750: procedure Khi2_Conform(N_cls    : Integer;
9751:                        N_estim  : Integer;
9752:                        Obs      : TIntVector;
9753:                        Calc     : TVector;
9754:                        var Khi2 : Float;
9755:                        var DoF  : Integer); external 'dmath';
9756: { Khi-2 test for conformity }
9757: procedure Khi2_Indep(N_lin    : Integer;
9758:                      N_col    : Integer;
9759:                      Obs      : TIntMatrix;
9760:                      var Khi2 : Float;
9761:                      var DoF  : Integer); external 'dmath';
9762: { Khi-2 test for independence }
9763: procedure Woolf_Conform(N_cls   : Integer;
9764:                         N_estim : Integer;
9765:                         Obs     : TIntVector;
9766:                         Calc    : TVector;
9767:                         var G   : Float;
9768:                         var DoF : Integer); external 'dmath';
9769: { Woolf's test for conformity }
9770: procedure Woolf_Indep(N_lin   : Integer;
9771:                       N_col   : Integer;
9772:                       Obs     : TIntMatrix;
9773:                       var G   : Float;
9774:                       var DoF : Integer); external 'dmath';
9775: { Woolf's test for independence }
9776: procedure DimStatClassVector(var C : TStatClassVector;
9777:                              Ub    : Integer); external 'dmath';
9778: { Allocates an array of statistical classes: C[0..Ub] }
9779: procedure Distrib(X       : TVector;
9780:                   Lb, Ub  : Integer;
9781:                   A, B, H : Float;
9782:                   C       : TStatClassVector); external 'dmath';
9783: { Distributes an array X[Lb..Ub] into statistical classes }
9784: { ---------------------------------------------------------------
9785:   Linear / polynomial regression
9786:   --------------------------------------------------------------- }
9787: procedure LinFit(X, Y   : TVector;
```

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

```
9877: { Sets the optimization algorithm for nonlinear regression }
9878: function GetOptAlgo : TOptAlgo; external 'dmath';
9879: { Returns the optimization algorithm }
9880: procedure SetMaxParam(N : Byte); external 'dmath';
9881: { Sets the maximum number of regression parameters for nonlinear regression }
9882: function GetMaxParam : Byte; external 'dmath';
9883: { Returns the maximum number of regression parameters for nonlinear regression }
9884: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9885: { Sets the bounds on the I-th regression parameter }
9886: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9887: { Returns the bounds on the I-th regression parameter }
9888: procedure NLFit(RegFunc   : TRegFunc;
9889:                 DerivProc : TDerivProc;
9890:                 X, Y      : TVector;
9891:                 Lb, Ub    : Integer;
9892:                 MaxIter   : Integer;
9893:                 Tol       : Float;
9894:                 B         : TVector;
9895:                 FirstPar,
9896:                 LastPar   : Integer;
9897:                 V         : TMatrix); external 'dmath';
9898: { Unweighted nonlinear regression }
9899: procedure WNLFit(RegFunc   : TRegFunc;
9900:                  DerivProc : TDerivProc;
9901:                  X, Y, S   : TVector;
9902:                  Lb, Ub    : Integer;
9903:                  MaxIter   : Integer;
9904:                  Tol       : Float;
9905:                  B         : TVector;
9906:                  FirstPar,
9907:                  LastPar   : Integer;
9908:                  V         : TMatrix); external 'dmath';
9909: { Weighted nonlinear regression }
9910: procedure SetMCFile(FileName : String); external 'dmath';
9911: { Set file for saving MCMC simulations }
9912: procedure SimFit(RegFunc   : TRegFunc;
9913:                  X, Y      : TVector;
9914:                  Lb, Ub    : Integer;
9915:                  B         : TVector;
9916:                  FirstPar,
9917:                  LastPar   : Integer;
9918:                  V         : TMatrix); external 'dmath';
9919: { Simulation of unweighted nonlinear regression by MCMC }
9920: procedure WSimFit(RegFunc   : TRegFunc;
9921:                   X, Y, S   : TVector;
9922:                   Lb, Ub    : Integer;
9923:                   B         : TVector;
9924:                   FirstPar,
9925:                   LastPar   : Integer;
9926:                   V         : TMatrix); external 'dmath';
9927: { Simulation of weighted nonlinear regression by MCMC }
9928: { ------------------------------------------------------------------
9929:   Nonlinear regression models
9930:   ------------------------------------------------------------------ }
9931: procedure FracFit(X, Y       : TVector;
9932:                   Lb, Ub     : Integer;
9933:                   Deg1, Deg2 : Integer;
9934:                   ConsTerm   : Boolean;
9935:                   MaxIter    : Integer;
9936:                   Tol        : Float;
9937:                   B          : TVector;
9938:                   V          : TMatrix); external 'dmath';
9939: { Unweighted fit of rational fraction }
9940: procedure WFracFit(X, Y, S    : TVector;
9941:                    Lb, Ub     : Integer;
9942:                    Deg1, Deg2 : Integer;
9943:                    ConsTerm   : Boolean;
9944:                    MaxIter    : Integer;
9945:                    Tol        : Float;
9946:                    B          : TVector;
9947:                    V          : TMatrix); external 'dmath';
9948: { Weighted fit of rational fraction }
9949:
9950: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9951: { Returns the value of the rational fraction at point X }
9952: procedure ExpFit(X, Y         : TVector;
9953:                  Lb, Ub, Nexp : Integer;
9954:                  ConsTerm     : Boolean;
9955:                  MaxIter      : Integer;
9956:                  Tol          : Float;
9957:                  B            : TVector;
9958:                  V            : TMatrix); external 'dmath';
9959: { Unweighted fit of sum of exponentials }
9960: procedure WExpFit(X, Y, S      : TVector;
9961:                   Lb, Ub, Nexp : Integer;
9962:                   ConsTerm     : Boolean;
9963:                   MaxIter      : Integer;
9964:                   Tol          : Float;
9965:                   B            : TVector;
```

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

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

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

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

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

```
10411: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10412: begin
10413:  Procedure ClearModules
10414:  Procedure ReadMapFile( Fname : string)
10415:  Function AddressInfo( Address : dword) : string
10416: end;
10417:
10418: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10419: begin
10420:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10421:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10422:    +'teByOther, tpExecuteByOther )
10423:   TTarPermissions', 'set of TTarPermission
10424:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10425:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10426:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10427:   TTarModes', 'set of TTarMode
10428:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10429:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10430:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10431:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10432:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10433:   SIRegister_TTarArchive(CL);
10434:   SIRegister_TTarWriter(CL);
10435:  Function PermissionString( Permissions : TTarPermissions) : STRING
10436:  Function ConvertFilename( Filename : STRING) : STRING
10437:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10438:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10439:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10440: end;
10441:
10442:
10443: //************************************************unit uPSI_TlHelp32;
10444: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10445: begin
10446:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10447:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10448:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10449:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10450:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10451:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10452:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10453:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10454:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10455:   AddTypeS('THeapList32', 'tagHEAPLIST32
10456:  Const('HF32_DEFAULT','LongInt'( 1);
10457:  Const('HF32_SHARED','LongInt'( 2);
10458:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10459:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10460:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10461:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10462:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10463:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10464:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10465:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10466:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10467:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10468:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10469:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10470:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10471:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10472:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10473:    +'aPri : Longint; dwFlags : DWORD; end
10474:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10475:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10476:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10477:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10478: end;
10479:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10480:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10481:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10482:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10483:  Const('EWX_LOGOFF','LongInt'( 0);
10484:  Const('EWX_SHUTDOWN','LongInt'( 1);
10485:  Const('EWX_REBOOT','LongInt'( 2);
10486:  Const('EWX_FORCE','LongInt'( 4);
10487:  Const('EWX_POWEROFF','LongInt'( 8);
10488:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10489:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10490:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10491:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10492:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10493:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10494:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10495:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10496:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10497:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10498:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10499:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
```

```
10500:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10501:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10502:  Function GetDesktopWindow : HWND
10503:  Function GetParent( hWnd : HWND) : HWND
10504:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10505:  Function GetTopWindow( hWnd : HWND) : HWND
10506:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10507:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10508:  //Delphi DFM
10509:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10510:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10511:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10512:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10513:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10514:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10515:  Function OpenIcon( hWnd : HWND) : BOOL
10516:  Function CloseWindow( hWnd : HWND) : BOOL
10517:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10518:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10519:  Function IsWindowVisible( hWnd : HWND) : BOOL
10520:  Function IsIconic( hWnd : HWND) : BOOL
10521:  Function AnyPopup : BOOL
10522:  Function BringWindowToTop( hWnd : HWND) : BOOL
10523:  Function IsZoomed( hWnd : HWND) : BOOL
10524:  Function IsWindow( hWnd : HWND) : BOOL
10525:  Function IsMenu( hMenu : HMENU) : BOOL
10526:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10527:  Function DestroyWindow( hWnd : HWND) : BOOL
10528:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10529:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10530:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10531:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10532:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10533:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10534:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10535:
10536: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10537: begin
10538:  const('ShowSetupDialogOptLong','String '--setup
10539:  PrimaryConfPathOptLong','String '--primary-config-path=
10540:  PrimaryConfPathOptShort','String '--pcp=
10541:  SecondaryConfPathOptLong','String '--secondary-config-path=
10542:  SecondaryConfPathOptShort','String '--scp=
10543:  NoSplashScreenOptLong','String '--no-splash-screen
10544:  NoSplashScreenOptShort','String '--nsc
10545:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10546:  SkipLastProjectOpt','String '--skip-last-project
10547:  DebugLogOpt','String '--debug-log=
10548:  DebugLogOptEnable','String '--debug-enable=
10549:  LanguageOpt','String '--language=
10550:  LazarusDirOpt','String '--lazarusdir=
10551:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10552:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10553:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10554:  Function IsHelpRequested : Boolean
10555:  Function IsVersionRequested : boolean
10556:  Function GetLanguageSpecified : string
10557:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10558:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10559:  Procedure ParseNoGuiCmdLineParams
10560:  Function ExtractCmdLineFilenames : TStrings
10561: end;
10562:
10563:
10564: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10565: begin
10566:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10567:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10568:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10569:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10570:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10571:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10572:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10573:  Function DirPathExists( DirectoryName : string) : boolean
10574:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10575:  Function ExtractFileNameOnly( const AFilename : string) : string
10576:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10577:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10578:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10579:  Function ForceDirectory( DirectoryName : string) : boolean
10580:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10581:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10582:  Function FileIsText( const AFilename : string) : boolean
10583:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10584:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10585:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10586:  Function TrimFilename( const AFilename : string) : string
10587:  Function ResolveDots( const AFilename : string) : string
10588:  Procedure ForcePathDelims( var FileName : string)
```

```
10589:  Function GetForcedPathDelims( const FileName : string) : String
10590:  Function CleanAndExpandFilename( const Filename : string) : string
10591:  Function CleanAndExpandDirectory( const Filename : string) : string
10592:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10593:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10594:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10595:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10596:  Function FileIsInPath( const Filename, Path : string) : boolean
10597:  Function AppendPathDelim( const Path : string) : string
10598:  Function ChompPathDelim( const Path : string) : string
10599:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10600:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10601:  Function MinimizeSearchPath( const SearchPath : string) : string
10602:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10603:  (*Function FileExistsUTF8( const Filename : string) : boolean
10604:  Function FileAgeUTF8( const FileName : string) : Longint
10605:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10606:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10607:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10608:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10609:  Procedure FindCloseUTF8( var F : TSearchrec)
10610:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10611:  Function FileGetAttrUTF8( const FileName : String) : Longint
10612:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10613:  Function DeleteFileUTF8( const FileName : String) : Boolean
10614:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10615:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10616:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10617:  Function GetCurrentDirUTF8 : String
10618:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10619:  Function CreateDirUTF8( const NewDir : String) : Boolean
10620:  Function RemoveDirUTF8( const Dir : String) : Boolean
10621:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10622:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10623:  Function FileCreateUTF8( const FileName : string) : THandle;
10624:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10625:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10626:  Function FileSizeUtf8( const Filename : string) : int64
10627:  Function GetFileDescription( const AFilename : string) : string
10628:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10629:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10630:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10631:  Function IsUNCPath( const Path : String) : Boolean
10632:  Function ExtractUNCVolume( const Path : String) : String
10633:  Function ExtractFileRoot( FileName : String) : String
10634:  Function GetDarwinSystemFilename( Filename : string) : string
10635:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10636:  Function StrToCmdLineParam( const Param : string) : string
10637:  Function MergeCmdLineParams( ParamList : TStrings) : string
10638:  Procedure InvalidateFileStateCache( const Filename : string)
10639:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10640:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10641:  Function FindAllDocs(const Root, extmask: string): TStringlist;
10642:  Function ReadFileToString( const Filename : string) : string
10643:  procedure Inc1(var X: longint; N: Longint);
10644:
10645:  type
10646:   TCopyFileFlag = ( cffOverwriteFile,
10647:     cffCreateDestDirectory, cffPreserveTime );
10648:   TCopyFileFlags = set of TCopyFileFlag;*)
10649:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10650:   TCopyFileFlags', 'set of TCopyFileFlag
10651:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10652: end;
10653:
10654: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10655: begin
10656:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10657:   SIRegister_TMask(CL);
10658:   SIRegister_TParseStringList(CL);
10659:   SIRegister_TMaskList(CL);
10660:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10661:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10662:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10663:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10664: end;
10665:
10666: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10667: begin
10668:   //PShellHookInfo', '^TShellHookInfo // will not work
10669:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10670:   SHELLHOOKINFO', 'TShellHookInfo
10671:   LPSHELLHOOKINFO', 'PShellHookInfo
10672:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10673:   SIRegister_TJvShellHook(CL);
10674:  Function InitJvShellHooks : Boolean
10675:  Procedure UnInitJvShellHooks
```

```
10676: end;
10677:
10678: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10679: begin
10680:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10681:    +', dcHasSetSel, dcWantTab, dcNative )
10682:   TDlgCodes', 'set of TDlgCode
10683:  'dcWantMessage',' dcWantAllKeys);
10684:   SIRegister_IJvExControl(CL);
10685:   SIRegister_IJvDenySubClassing(CL);
10686:   SIRegister_TStructPtrMessage(CL);
10687:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10688:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10689:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10690:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10691:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10692:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10693:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10694:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10695:  Function GetFocusedControl( AControl : TControl) : TWinControl
10696:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10697:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10698:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10699:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10700:   SIRegister_TJvExControl(CL);
10701:   SIRegister_TJvExWinControl(CL);
10702:   SIRegister_TJvExCustomControl(CL);
10703:   SIRegister_TJvExGraphicControl(CL);
10704:   SIRegister_TJvExHintWindow(CL);
10705:   SIRegister_TJvExPubGraphicControl(CL);
10706: end;
10707:
10708: (*----------------------------------------------------------------------------*)
10709: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10710: begin
10711:  Procedure EncodeStream( Input, Output : TStream)
10712:  Procedure DecodeStream( Input, Output : TStream)
10713:  Function EncodeString1( const Input : string) : string
10714:  Function DecodeString1( const Input : string) : string
10715: end;
10716:
10717: (*----------------------------------------------------------------------------*)
10718: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10719: begin
10720:   SIRegister_TWebAppRegInfo(CL);
10721:   SIRegister_TWebAppRegList(CL);
10722:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10723:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10724:  Procedure UnregisterWebApp( const AProgID : string)
10725:  Function FindRegisteredWebApp( const AProgID : string) : string
10726:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10727:  'sUDPPort','String 'UDPPort
10728: end;
10729:
10730: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10731: begin
10732:  // TStringDynArray', 'array of string
10733:  Function GetEnvVarValue( const VarName : string) : string
10734:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10735:  Function DeleteEnvVar( const VarName : string) : Integer
10736:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10737:  Function ExpandEnvVars( const Str : string) : string
10738:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10739:  Procedure GetAllEnvVarNames( const Names : TStrings);
10740:  Function GetAllEnvVarNames1 : TStringDynArray;
10741:  Function EnvBlockSize : Integer
10742:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10743:   SIRegister_TPJEnvVarsEnumerator(CL);
10744:   SIRegister_TPJEnvVars(CL);
10745:   FindClass('TOBJECT'),'EPJEnvVars
10746:   FindClass('TOBJECT'),'EPJEnvVars
10747:  //Procedure Register
10748: end;
10749:
10750: (*----------------------------------------------------------------------------*)
10751: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10752: begin
10753:  'cOneSecInMS','LongInt'( 1000);
10754:  //'cDefTimeSlice','LongInt'( 50);
10755:  //'cDefMaxExecTime',' cOneMinInMS);
10756:  'cAppErrorMask','LongInt'( 1 shl 29);
10757:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10758:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10759:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10760:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10761:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10762:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10763:  Function MakeSize( const ACX, ACY : LongInt) : TSize
```

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

```
10853:
10854: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10855: begin
10856:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10857:   SIRegister_TZSortedList(CL);
10858:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10859:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10860:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10861:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10862:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10863:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10864:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10865:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10866:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10867:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10868:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10869:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10870:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10871:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10872:  Function StrToBoolEx( Str : string) : Boolean
10873:  Function BoolToStrEx( Bool : Boolean) : String
10874:  Function IsIpAddr( const Str : string) : Boolean
10875:  Function zSplitString( const Str, Delimiters : string) : TStrings
10876:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10877:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10878:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10879:  Function FloatToSQLStr( Value : Extended) : string
10880:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10881:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10882:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10883:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10884:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10885:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10886:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10887:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10888:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10889:  Function BytesToVar( const Value : TByteDynArray) : Variant
10890:  Function VarToBytes( const Value : Variant) : TByteDynArray
10891:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10892:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10893:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10894:  Function EncodeCString( const Value : string) : string
10895:  Function DecodeCString( const Value : string) : string
10896:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10897:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10898:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10899:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10900:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10901:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10902:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10903:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10904:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10905:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10906:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10907:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10908:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10909:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10910: end;
10911:
10912: unit uPSI_ZEncoding;
10913:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10914:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10915:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10916:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10917:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10918:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10919:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10920:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10921:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10922:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10923:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10924:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10925:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10926:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10927:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10928:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10929:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10930:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10931:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10932:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10933:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10934:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10935:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10936:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10937:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10938:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10939:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
```

```
10940:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10941:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10942:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10943:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10944:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10945:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10946:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10947:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10948:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10949:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10950:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10951:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10952:  Function ZDefaultSystemCodePage : Word
10953:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10954:
10955:
10956: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10957: begin
10958:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10959:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10960:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10961:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10962:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10963:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10964:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10965:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10966:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10967:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10968:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10969:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10970:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10971:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10972:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10973:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10974:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10975:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10976:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10977:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10978:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10979:  ('ilIdentity','2 RPC_C_IMP_LEVEL_IDENTIFY);
10980:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10981:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10982:  ('EOAC_NONE','LongWord').SetUInt( $0);
10983:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10984:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10985:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10986:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10987:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10988:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10989:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10990:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10991:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10992:   FindClass('TOBJECT'),'EBoldCom
10993:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10994:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10995:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10996:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10997:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10998:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10999:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
11000:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
11001:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11002:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
11003:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
11004:  Function BoldCreateGUID : TGUID
11005:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
11006:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
11007:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11008:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11009: end;
11010:
11011: (*----------------------------------------------------------------------------*)
11012: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11013: begin
11014:  Function ParseISODate( s : string) : TDateTime
11015:  Function ParseISODateTime( s : string) : TDateTime
11016:  Function ParseISOTime( str : string) : TDateTime
11017: end;
11018:
11019: (*----------------------------------------------------------------------------*)
11020: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11021: begin
11022:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
11023:  Function BoldCreateGUIDWithBracketsAsString : string
11024: end;
11025:
11026: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11027: begin
```

```
11028:   FindClass('TOBJECT'),'TBoldFileHandler
11029:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11030:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11031:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
11032:   SIRegister_TBoldFileHandler(CL);
11033:   SIRegister_TBoldDiskFileHandler(CL);
11034:  Procedure BoldCloseAllFilehandlers
11035:  Procedure BoldRemoveUnchangedFilesFromEditor
11036:  Function BoldFileHandlerList : TBoldObjectArray
11037:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11038: end;
11039:
11040: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11041: begin
11042:   PCharArr', 'array of PChar
11043:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
11044:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11045:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11046:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11047:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
11048:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
11049:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
11050:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11051:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11052:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11053:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11054:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11055: end;
11056:
11057: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11058: begin
11059:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11060:   SIRegister_TfrmBoldQueryUser(CL);
11061:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11062: end;
11063:
11064: (*----------------------------------------------------------------------------*)
11065: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11066: begin
11067:  //('befIsInDisplayList',' BoldElementFlag0);
11068:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11069:  //('befFollowerSelected',' BoldElementFlag2);
11070:   FindClass('TOBJECT'),'TBoldQueue
11071:   FindClass('TOBJECT'),'TBoldQueueable
11072:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11073:   SIRegister_TBoldQueueable(CL);
11074:   SIRegister_TBoldQueue(CL);
11075:  Function BoldQueueFinalized : Boolean
11076:  Function BoldInstalledQueue : TBoldQueue
11077: end;
11078:
11079: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11080: begin
11081:  const mmPerInch','Extended').setExtended( 25.4);
11082:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11083:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11084:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11085:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11086:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11087:   TBarLineType', '( white, black, black_half )
11088:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11089:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11090:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11091:   TCheckSumMethod', '( csmNone, csmModulo10 )
11092:   SIRegister_TAsBarcode(CL);
11093:  Function CheckSumModulo10( const data : string) : string
11094:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11095:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11096:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11097:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11098: end;
11099:
11100: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11101: begin
11102:    THomogeneousByteVector', 'array[0..3] of Byte
11103:   THomogeneousWordVector', 'array[0..3] of Word
11104:   THomogeneousIntVector', 'array[0..3] of Integer
11105:   THomogeneousFltVector', 'array[0..3] of single
11106:   THomogeneousDblVector', 'array[0..3] of double
11107:   THomogeneousExtVector', 'array[0..3] of extended
11108:   TAffineByteVector', 'array[0..2] of Byte
11109:   TAffineWordVector', 'array[0..2] of Word
```

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

```
11199:  Function MakeVector( V : array of Single) : TVectorGL
11200:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11201:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11202:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11203:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11204:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11205:  Function ArcCosGL( X : Extended) : Extended
11206:  Function ArcSinGL( X : Extended) : Extended
11207:  Function ArcTan2GL( Y, X : Extended) : Extended
11208:  Function CoTanGL( X : Extended) : Extended
11209:  Function DegToRadGL( Degrees : Extended) : Extended
11210:  Function RadToDegGL( Radians : Extended) : Extended
11211:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11212:  Function TanGL( X : Extended) : Extended
11213:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11214:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11215:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11216:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11217:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11218:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11219: end;
11220:
11221:
11222: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11223: begin
11224:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11225:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11226:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11227:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11228:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11229:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11230:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11231:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11232:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11233:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11234:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11235:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11236:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11237:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11238:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11239:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11240:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11241:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11242:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11243:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11244:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11245:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11246:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11247:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11248:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11249:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11250:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11251: end;
11252:
11253: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11254: begin
11255:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11256:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11257:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11258:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11259:   FindClass('TOBJECT'),'EInvalidParam
11260:  Function IsDCOMInstalled : Boolean
11261:  Function IsDCOMEnabled : Boolean
11262:  Function GetDCOMVersion : string
11263:  Function GetMDACVersion : string
11264:  Function GetMDACVersion2 : string
11265:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11266:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11267:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11268:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11269:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11270:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11271:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11272:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11273:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11274:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11275:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11276:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11277:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11278:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11279: end;
11280:
11281:
11282: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
```

```
11283: begin
11284:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11285:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11286:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11287:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11288:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11289:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11290:  DegPerCycle','Extended').setExtended( 360.0);
11291:  DegPerGrad','Extended').setExtended( 0.9);
11292:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11293:  GradPerCycle','Extended').setExtended( 400.0);
11294:  GradPerDeg','Extended').setExtended( 1.1111111111111111111111111111111);
11295:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11296:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11297:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11298:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11299:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11300:  CyclePerGrad','Extended').setExtended( 0.0025);
11301:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11302:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11303:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11304:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11305:  Function MakePercentage( const Step, Max : Longint) : Longint
11306:  Function CelsiusToKelvin( const T : double) : double
11307:  Function CelsiusToFahrenheit( const T : double) : double
11308:  Function KelvinToCelsius( const T : double) : double
11309:  Function KelvinToFahrenheit( const T : double) : double
11310:  Function FahrenheitToCelsius( const T : double) : double
11311:  Function FahrenheitToKelvin( const T : double) : double
11312:  Function CycleToDeg( const Cycles : double) : double
11313:  Function CycleToGrad( const Cycles : double) : double
11314:  Function CycleToRad( const Cycles : double) : double
11315:  Function DegToCycle( const Degrees : double) : double
11316:  Function DegToGrad( const Degrees : double) : double
11317:  Function DegToRad( const Degrees : double) : double
11318:  Function GradToCycle( const Grads : double) : double
11319:  Function GradToDeg( const Grads : double) : double
11320:  Function GradToRad( const Grads : double) : double
11321:  Function RadToCycle( const Radians : double) : double
11322:  Function RadToDeg( const Radians : double) : double
11323:  Function RadToGrad( const Radians : double) : double
11324:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11325:  Function DmsToRad( const D, M : Integer; const S : double) : double
11326:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11327:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11328:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11329:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11330:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11331:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11332:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11333:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11334:  Function CmToInch( const Cm : double) : double
11335:  Function InchToCm( const Inch : double) : double
11336:  Function FeetToMetre( const Feet : double) : double
11337:  Function MetreToFeet( const Metre : double) : double
11338:  Function YardToMetre( const Yard : double) : double
11339:  Function MetreToYard( const Metre : double) : double
11340:  Function NmToKm( const Nm : double) : double
11341:  Function KmToNm( const Km : double) : double
11342:  Function KmToSm( const Km : double) : double
11343:  Function SmToKm( const Sm : double) : double
11344:  Function LitreToGalUs( const Litre : double) : double
11345:  Function GalUsToLitre( const GalUs : double) : double
11346:  Function GalUsToGalCan( const GalUs : double) : double
11347:  Function GalCanToGalUs( const GalCan : double) : double
11348:  Function GalUsToGalUk( const GalUs : double) : double
11349:  Function GalUkToGalUs( const GalUk : double) : double
11350:  Function LitreToGalCan( const Litre : double) : double
11351:  Function GalCanToLitre( const GalCan : double) : double
11352:  Function LitreToGalUk( const Litre : double) : double
11353:  Function GalUkToLitre( const GalUk : double) : double
11354:  Function KgToLb( const Kg : double) : double
11355:  Function LbToKg( const Lb : double) : double
11356:  Function KgToOz( const Kg : double) : double
11357:  Function OzToKg( const Oz : double) : double
11358:  Function CwtUsToKg( const Cwt : double) : double
11359:  Function CwtUkToKg( const Cwt : double) : double
11360:  Function KaratToKg( const Karat : double) : double
11361:  Function KgToCwtUs( const Kg : double) : double
11362:  Function KgToCwtUk( const Kg : double) : double
11363:  Function KgToKarat( const Kg : double) : double
11364:  Function KgToSton( const Kg : double) : double
11365:  Function KgToLton( const Kg : double) : double
11366:  Function StonToKg( const STon : double) : double
11367:  Function LtonToKg( const Lton : double) : double
11368:  Function QrUsToKg( const Qr : double) : double
11369:  Function QrUkToKg( const Qr : double) : double
11370:  Function KgToQrUs( const Kg : double) : double
11371:  Function KgToQrUk( const Kg : double) : double
```

```
11372:  Function PascalToBar( const Pa : double) : double
11373:  Function PascalToAt( const Pa : double) : double
11374:  Function PascalToTorr( const Pa : double) : double
11375:  Function BarToPascal( const Bar : double) : double
11376:  Function AtToPascal( const At : double) : double
11377:  Function TorrToPascal( const Torr : double) : double
11378:  Function KnotToMs( const Knot : double) : double
11379:  Function HpElectricToWatt( const HpE : double) : double
11380:  Function HpMetricToWatt( const HpM : double) : double
11381:  Function MsToKnot( const ms : double) : double
11382:  Function WattToHpElectric( const W : double) : double
11383:  Function WattToHpMetric( const W : double) : double
11384:  function getBigPI: string;    //PI of 1000 numbers
11385:
11386: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11387: begin
11388:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11389:  Procedure CDCopyFile( const FileName, DestName : string)
11390:  Procedure CDMoveFile( const FileName, DestName : string)
11391:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11392:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11393:  Function CDGetTempDir : string
11394:  Function CDGetFileSize( FileName : string) : longint
11395:  Function GetFileTime( FileName : string) : longint
11396:  Function GetShortName( FileName : string) : string
11397:  Function GetFullName( FileName : string) : string
11398:  Function WinReboot : boolean
11399:  Function WinDir : String
11400:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11401:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11402:  Function devExecutor : TdevExecutor
11403: end;
11404:
11405: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11406: begin
11407:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11408:  Procedure Associate( Index : integer)
11409:  Procedure UnAssociate( Index : integer)
11410:  Function IsAssociated( Index : integer) : boolean
11411:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11412:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11413:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11414:  procedure RefreshIcons;
11415: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11416: function MergColor(Colors: Array of TColor): TColor;
11417: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11418: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11419: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11420: function GetInverseColor(AColor: TColor): TColor;
11421: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11422: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11423: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11424: Procedure GetSystemMenuFont(Font: TFont);
11425: end;
11426:
11427: //*************************unit uPSI_JvHLParser;*******************************
11428: function IsStringConstant(const St: string): Boolean;
11429: function IsIntConstant(const St: string): Boolean;
11430: function IsRealConstant(const St: string): Boolean;
11431: function IsIdentifier(const ID: string): Boolean;
11432: function GetStringValue(const St: string): string;
11433: procedure ParseString(const S: string; Ss: TStrings);
11434: function IsStringConstantW(const St: WideString): Boolean;
11435: function IsIntConstantW(const St: WideString): Boolean;
11436: function IsRealConstantW(const St: WideString): Boolean;
11437: function IsIdentifierW(const ID: WideString): Boolean;
11438: function GetStringValueW(const St: WideString): WideString;
11439: procedure ParseStringW(const S: WideString; Ss: TStrings);
11440:
11441:
11442: //*************************unit uPSI_JclMapi;*******************************
11443:
11444: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11445: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11446: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11447: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11448: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11449:
11450: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11451: begin
11452:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11453:  Function BuildType1Message( ADomain, AHost : String) : String
11454:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11455:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11456:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11457: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

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

```
11546:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11547:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11548:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11549: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11550:   SIRegister_TIdSysLogMsgPart(CL);
11551:   SIRegister_TIdSysLogMessage(CL);
11552:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11553:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11554:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11555:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11556:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11557:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11558: end;
11559:
11560: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11561: begin
11562:  'UWhitespace','String '(?:\s*)
11563:  Function StripSpaces( const AText : string) : string
11564:  Function CharCount( const AText : string; Ch : Char) : Integer
11565:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11566:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11567: end;
11568:
11569:
11570: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11571: begin
11572:  ExtPascalVersion','String '0.9.8
11573:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11574:    +'Opera, brKonqueror, brMobileSafari )
11575:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11576:   AddTypeS('TExtProcedure', 'Procedure
11577:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11578:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11579:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11580:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11581:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11582:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11583:  Function StrToJS( const S : string; UseBR : boolean) : string
11584:  Function CaseOf( const S : string; const Cases : array of string) : integer
11585:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11586:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11587:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11588:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11589:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11590:  Function IsUpperCase( S : string) : boolean
11591:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11592:  Function BeautifyCSS( const AStyle : string) : string
11593:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11594:  Function JSDateToDateTime( JSDate : string) : TDateTime
11595: end;
11596:
11597: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11598: begin
11599:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11600:   TSHDeleteOptions', 'set of TSHDeleteOption
11601:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11602:   TSHRenameOptions', 'set of TSHRenameOption
11603:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11604:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11605:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11606:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11607:   TEnumFolderFlags', 'set of TEnumFolderFlag
11608:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11609:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11610:    +'IEnumIdList; Folder : IShellFolder; end
11611:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11612:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11613:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11614:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11615:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11616:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11617:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11618:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11619:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11620:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11621:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11622:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11623:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11624:  Function SHFreeMem( var P : Pointer) : Boolean
11625:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11626:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11627:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11628:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11629:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11630:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11631:  Function PidlFree( var IdList : PItemIdList) : Boolean
11632:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11633:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11634:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
```

```
11635:  Function PidlToPath( IdList : PItemIdList) : string
11636:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11637:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11638:   PShellLink', '^TShellLink // will not work
11639:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11640:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11641:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11642:  Procedure ShellLinkFree( var Link : TShellLink)
11643:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11644:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11645:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11646:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11647:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11648:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11649:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11650:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11651:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11652:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11653:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11654:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11655:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11656:  Function ShellOpenAs( const FileName : string) : Boolean
11657:  Function ShellRasDial( const EntryName : string) : Boolean
11658:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11659:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11660:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11661:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11662:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11663:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11664:  Function OemKeyScan( wOemChar : Word) : DWORD
11665:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11666: end;
11667:
11668: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11669: begin
11670:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11671:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11672:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11673:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11674:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11675:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11676:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11677:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11678:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11679:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11680:  Function xmlValidName( const Text : UnicodeString) : Boolean
11681:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11682:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11683:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11684:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11685:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
     : TUnicodeCodecClass
11686:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11687:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11688:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11689:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11690:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11691:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11692:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11693:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11694:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11695:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11696:  Procedure SelfTestcXMLFunctions
11697: end;
11698:
11699: (*----------------------------------------------------------------------------*)
11700: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11701: begin
11702:  Function AWaitCursor : IUnknown
11703:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11704:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11705:  Function YesNo( const ACaption, AMsg : string) : boolean
11706:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11707:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11708:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11709:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11710:  Procedure GetSystemPaths( Strings : TStrings)
11711:  Procedure MakeEditNumeric( EditHandle : integer)
11712: end;
11713:
11714: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11715: begin
11716:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11717:  'BI_YUY2','LongWord( $32595559);
11718:  'BI_UYVY','LongWord').SetUInt( $59565955);
11719:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11720:  'BI_YVU9','LongWord').SetUInt( $39555659);
11721:  'BI_YUV12','LongWord( $30323449);
11722:  'BI_Y8','LongWord').SetUInt( $20203859);
```

```
11723:  'BI_Y211','LongWord').SetUInt( $31313259);
11724:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11725:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11726: end;
11727:
11728: (*----------------------------------------------------------------------------*)
11729: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11730: begin
11731:  'WM_USER','LongWord').SetUInt( $0400);
11732:  'WM_CAP_START','LongWord').SetUint($0400);
11733:  'WM_CAP_END','longword').SetUint($0400+85);
11734:  //WM_CAP_START+  85
11735:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11736:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11737:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11738:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11739:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11740:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11741:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11742:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11743:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11744:  Function capGetUserData( hwnd : THandle) : LongInt
11745:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11746:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11747:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11748:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11749:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11750:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11751:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11752:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11753:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11754:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11755:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11756:  Function capEditCopy( hwnd : THandle) : LongInt
11757:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11758:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11759:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11760:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11761:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11762:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11763:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11764:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11765:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11766:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11767:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11768:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11769:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11770:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11771:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11772:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11773:  Function capGrabFrame( hwnd : THandle) : LongInt
11774:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11775:  Function capCaptureSequence( hwnd : THandle) : LongInt
11776:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11777:  Function capCaptureStop( hwnd : THandle) : LongInt
11778:  Function capCaptureAbort( hwnd : THandle) : LongInt
11779:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11780:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11781:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11782:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11783:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11784:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11785:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11786:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11787:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11788:  Function capPalettePaste( hwnd : THandle) : LongInt
11789:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11790:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11791:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11792:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11793:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11794:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11795:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11796:   //PCapStatus', '^TCapStatus // will not work
11797:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11798:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11799:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11800:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11801:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11802:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11803:    +' wNumAudioAllocated : WORD; end
11804:   //PCaptureParms', '^TCaptureParms // will not work
11805:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11806:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11807:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11808:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11809:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11810:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11811:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
```

```
11812:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11813:    +'he : BOOL; AVStreamMaster : WORD; end
11814:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11815:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11816:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11817:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11818:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
       : Integer; hwndParent : THandle; nID : Integer) : THandle
11819:  Function
       capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11820:  'IDS_CAP_BEGIN','LongInt'( 300);
11821:  'IDS_CAP_END','LongInt'( 301);
11822:  'IDS_CAP_INFO','LongInt'( 401);
11823:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11824:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11825:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11826:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11827:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11828:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11829:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11830:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11831:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11832:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11833:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11834:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11835:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11836:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11837:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11838:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11839:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11840:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11841:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11842:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11843:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11844:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11845:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11846:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11847:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11848:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11849:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11850:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11851:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11852:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11853:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11854:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11855:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11856:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11857:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11858:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11859:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11860:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11861:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11862:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11863:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11864:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11865:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11866:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11867:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11868:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11869:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11870:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11871:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11872:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11873:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11874:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11875:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11876:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11877:  'AVICAP32','String 'AVICAP32.dll
11878: end;
11879:
11880: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11881: begin
11882:  Function AlBoolToInt( Value : Boolean) : Integer
11883:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11884:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11885:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11886:  Function ALInc( var x : integer; Count : integer) : Integer
11887:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11888:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11889:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11890:  Function ALIsInteger(const S: AnsiString): Boolean;
11891:  function ALIsDecimal(const S: AnsiString): boolean;
11892:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11893:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11894:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11895:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11896:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11897:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11898:  Function ALRandomStr(const aLength: Longint): AnsiString;
```

```
11899:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11900:  Function ALRandomStrU(const aLength: Longint): String;
11901: end;
11902:
11903: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11904: begin
11905:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11906: end;
11907:
11908: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11909: begin
11910:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11911:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11912:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11913:    +'; ullAvailExtendedVirtual : Int64; end
11914:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11915:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11916:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11917:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11918:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11919:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11920:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11921:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11922: end;
11923:
11924: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11925: begin
11926:   SIRegister_THandledObject(CL);
11927:   SIRegister_TEvent(CL);
11928:   SIRegister_TMutex(CL);
11929:   SIRegister_TSharedMem(CL);
11930:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11931:  'TRACE_BUFFER','String 'TRACE_BUFFER
11932:  'TRACE_MUTEX','String 'TRACE_MUTEX
11933:   //PTraceEntry', '^TTraceEntry // will not work
11934:   SIRegister_TIPCTracer(CL);
11935:  'MAX_CLIENTS','LongInt'( 6);
11936:  'IPCTIMEOUT','LongInt'( 2000);
11937:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11938:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11939:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11940:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11941:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11942:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11943:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11944:   FindClass('TOBJECT'),'EMonitorActive
11945:   FindClass('TOBJECT'),'TIPCThread
11946:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11947:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11948:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11949:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11950:   TClientFlags', 'set of TClientFlag
11951:   //PEventData', '^TEventData // will not work
11952:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11953:    +'lag; Flags : TClientFlags; end
11954:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11955:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11956:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11957:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11958:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11959:   SIRegister_TIPCEvent(CL);
11960:   //PClientDirRecords', '^TClientDirRecords // will not work
11961:   SIRegister_TClientDirectory(CL);
11962:   TIPCState', '( stInActive, stDisconnected, stConnected )
11963:   SIRegister_TIPCThread(CL);
11964:   SIRegister_TIPCMonitor(CL);
11965:   SIRegister_TIPCClient(CL);
11966:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11967: end;
11968:
11969: (*----------------------------------------------------------------------------*)
11970: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11971: begin
11972:   SIRegister_TAlGSMComm(CL);
11973:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11974:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
       AMessage:AnsiString);
11975:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11976:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
       UseGreekAlphabet:Bool):Widestring;
11977:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11978:  end;
11979:
11980: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11981: begin
11982:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11983:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11984:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
```

```
11985:   TInternetScheme', 'integer
11986:   TALIPv6Binary', 'array[1..16] of Char;
11987:  // TALIPv6Binary = array[1..16] of ansiChar;
11988:  //   TInternetScheme = Integer;
11989:   SIRegister_TALHTTPRequestHeader(CL);
11990:   SIRegister_TALHTTPCookie(CL);
11991:   SIRegister_TALHTTPCookieCollection(CL);
11992:   SIRegister_TALHTTPResponseHeader(CL);
11993:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11994:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11995: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
       Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11996: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
       Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11997: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11998:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11999:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
12000:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
12001:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
12002:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
12003:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
       ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
12004:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
       Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
12005:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
12006:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
12007:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
12008:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
12009:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
12010:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
12011:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
12012:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
12013:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
12014:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
12015:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
12016:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
12017:  Function ALZeroIpV6 : TALIPv6Binary
12018:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
12019:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
12020:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
12021:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
12022: end;
12023:
12024: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
12025: begin
12026:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
12027:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12028:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
12029:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12030:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
12031:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
12032:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
12033:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
12034:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
12035:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12036:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
12037: end;
12038:
12039: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12040: begin
12041:   SIRegister_TALEMailHeader(CL);
12042:   SIRegister_TALNewsArticleHeader(CL);
12043:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
12044:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
12045:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
12046:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
12047:  Function AlGenerateInternetMessageID : AnsiString;
12048:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
12049:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
12050:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12051: end;
12052:
12053: (*----------------------------------------------------------------------------*)
12054: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12055: begin
12056:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12057:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
12058:  Function ALgetLocalIPs : TALStrings
12059:  Function ALgetLocalHostName : AnsiString
12060: end;
12061:
12062: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12063: begin
12064:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
```

```
12065:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12066:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12067:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12068:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12069:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12070:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12071:   +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12072:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
12073: end;
12074:
12075: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12076: begin
12077:   TStartupInfoA', 'TStartupInfo
12078:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12079:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12080:  SE_LOCK_MEMORY_NAME','String')( 'SeLockMemoryPrivilege
12081:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12082:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12083:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12084:  SE_TCB_NAME','String 'SeTcbPrivilege
12085:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12086:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12087:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12088:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12089:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12090:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12091:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12092:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12093:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12094:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12095:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12096:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12097:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12098:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12099:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12100:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12101:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12102:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12103:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12104:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12105:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12106:  Function AlGetEnvironmentString : AnsiString
12107:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12108:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12109:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12110:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12111:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12112: end;
12113:
12114: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12115: begin
12116:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12117:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12118:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12119:  Function ALGetModuleName : ansistring
12120:  Function ALGetModuleFileNameWithoutExtension : ansistring
12121:  Function ALGetModulePath : ansiString
12122:  Function AlGetFileSize( const AFileName : ansistring) : int64
12123:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12124:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12125:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12126:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12127:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12128:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12129:  Function ALFileExists( const Path : ansiString) : boolean
12130:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12131:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12132:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12133:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12134:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12135: end;
12136:
12137: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12138: begin
12139:   NativeInt', 'Integer
12140:   NativeUInt', 'Cardinal
```

```
12141:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12142:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12143:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12144:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12145:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12146:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12147:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12148:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12149:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12150:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12151:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12152:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12153:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12154:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12155:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12156:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12157:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12158:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12159:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12160:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12161:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12162:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12163:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12164:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12165:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12166:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12167:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12168:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12169:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12170:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12171:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12172:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12173: end;
12174:
12175: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12176: begin
12177:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12178:   FindClass('TOBJECT'),'TALXMLNode
12179:   FindClass('TOBJECT'),'TALXMLNodeList
12180:   FindClass('TOBJECT'),'TALXMLDocument
12181:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12182:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12183:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12184:    +'nst Name : AnsiString; const Attributes : TALStrings)
12185:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12186:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12187:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12188:    +'ntDocType, ntDocFragment, ntNotation )
12189:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12190:   TALXMLDocOptions', 'set of TALXMLDocOption
12191:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12192:   TALXMLParseOptions', 'set of TALXMLParseOption
12193:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12194:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12195:   SIRegister_EALXMLDocError(CL);
12196:   SIRegister_TALXMLNodeList(CL);
12197:   SIRegister_TALXMLNode(CL);
12198:   SIRegister_TALXmlElementNode(CL);
12199:   SIRegister_TALXmlAttributeNode(CL);
12200:   SIRegister_TALXmlTextNode(CL);
12201:   SIRegister_TALXmlDocumentNode(CL);
12202:   SIRegister_TALXmlCommentNode(CL);
12203:   SIRegister_TALXmlProcessingInstrNode(CL);
12204:   SIRegister_TALXmlCDataNode(CL);
12205:   SIRegister_TALXmlEntityRefNode(CL);
12206:   SIRegister_TALXmlEntityNode(CL);
12207:   SIRegister_TALXmlDocTypeNode(CL);
12208:   SIRegister_TALXmlDocFragmentNode(CL);
12209:   SIRegister_TALXmlNotationNode(CL);
12210:   SIRegister_TALXMLDocument(CL);
12211:  cAlXMLUTF8EncodingStr','String 'UTF-8
12212:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12213:  CALNSDelim','String ':
12214:  CALXML','String 'xml
12215:  CALVersion','String 'version
12216:  CALEncoding','String 'encoding
12217:  CALStandalone','String 'standalone
```

```
12218:  CALDefaultNodeIndent','String '
12219:  CALXmlDocument','String 'DOCUMENT
12220:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12221:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
        EncodingStr:AnsiString);
12222:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
        ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12223:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
        ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12224:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
        Recurse: Boolean):TalxmlNode
12225:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
        AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12226:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
        AnsiString
12227: end;
12228:
12229: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12230: //based on TEEProc, TeCanvas, TEEngine, TChart
12231: begin
12232:  'TeePiStep','Double').setExtended( Pi / 180.0);
12233:  'TeeDefaultPerspective','LongInt'( 100);
12234:  'TeeMinAngle','LongInt'( 270);
12235:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12236:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12237:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12238:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12239:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12240:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12241:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12242:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12243:  'TA_LEFT','LongInt'( 0);
12244:  'TA_RIGHT','LongInt'( 2);
12245:  'TA_CENTER','LongInt'( 6);
12246:  'TA_TOP','LongInt'( 0);
12247:  'TA_BOTTOM','LongInt'( 8);
12248:  'teePATCOPY','LongInt'( 0);
12249:  'NumCirclePoints','LongInt'( 64);
12250:  'teeDEFAULT_CHARSET','LongInt'( 1);
12251:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12252:  'TA_LEFT','LongInt'( 0);
12253:  'bs_Solid','LongInt'( 0);
12254:  'teepf24Bit','LongInt'( 0);
12255:  'teepfDevice','LongInt'( 1);
12256:  'CM_MOUSELEAVE','LongInt'( 10000);
12257:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12258:  'DC_BRUSH','LongInt'( 18);
12259:  'DC_PEN','LongInt'( 19);
12260:   teeCOLORREF', 'LongWord
12261:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12262:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12263:   SIRegister_TFilterRegion(CL);
12264:   SIRegister_IFormCreator(CL);
12265:   SIRegister_TTeeFilter(CL);
12266:   //TFilterClass', 'class of TTeeFilter
12267:   SIRegister_TFilterItems(CL);
12268:   SIRegister_TConvolveFilter(CL);
12269:   SIRegister_TBlurFilter(CL);
12270:   SIRegister_TTeePicture(CL);
12271:   TPenEndStyle', '( esRound, esSquare, esFlat )
12272:   SIRegister_TChartPen(CL);
12273:   SIRegister_TChartHiddenPen(CL);
12274:   SIRegister_TDottedGrayPen(CL);
12275:   SIRegister_TDarkGrayPen(CL);
12276:   SIRegister_TWhitePen(CL);
12277:   SIRegister_TChartBrush(CL);
12278:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12279:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12280:   SIRegister_TView3DOptions(CL);
12281:   FindClass('TOBJECT'),'TTeeCanvas
12282:   TTeeTransparency', 'Integer
12283:   SIRegister_TTeeBlend(CL);
12284:   FindClass('TOBJECT'),'TCanvas3D
12285:   SIRegister_TTeeShadow(CL);
12286:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
        gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12287:   FindClass('TOBJECT'),'TSubGradient
12288:   SIRegister_TCustomTeeGradient(CL);
12289:   SIRegister_TSubGradient(CL);
12290:   SIRegister_TTeeGradient(CL);
12291:   SIRegister_TTeeFontGradient(CL);
12292:   SIRegister_TTeeFont(CL);
12293:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12294:   TCanvasTextAlign', 'Integer
12295:   TTeeCanvasHandle', 'HDC
12296:   SIRegister_TTeeCanvas(CL);
12297:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12298:   SIRegister_TFloatXYZ(CL);
12299:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
```

```
12300:   TRGB', 'record blue: byte; green: byte; red: byte; end
12301:   {TRGB=packed record
12302:     Blue  : Byte;
12303:     Green : Byte;
12304:     Red   : Byte;
12305:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12306:
12307: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12308:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12309: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12310: TCanvas3DPlane', '( cpX, cpY, cpZ )
12311:   //IInterface', 'IUnknown
12312:   SIRegister_TCanvas3D(CL);
12313:   SIRegister_TTeeCanvas3D(CL);
12314:   TTrianglePoints', 'Array[0..2] of TPoint;
12315:   TFourPoints', 'Array[0..3] of TPoint;
12316:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12317:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12318:  Function Point3D( const x, y, z : Integer) : TPoint3D
12319:  Procedure SwapDouble( var a, b : Double)
12320:  Procedure SwapInteger( var a, b : Integer)
12321:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12322:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12323:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12324:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12325:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12326:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12327:  Procedure UnClipCanvas( ACanvas : TCanvas)
12328:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12329:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12330:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12331:  'TeeCharForHeight','String 'W
12332:  'DarkerColorQuantity','Byte').SetUInt( 128);
12333:  'DarkColorQuantity','Byte').SetUInt( 64);
12334:   TButtonGetColorProc', 'Function  : TColor
12335:   SIRegister_TTeeButton(CL);
12336:   SIRegister_TButtonColor(CL);
12337:   SIRegister_TComboFlat(CL);
12338:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12339:  Function TeePoint( const aX, aY : Integer) : TPoint
12340:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12341:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12342:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12343:  Function OrientRectangle( const R : TRect) : TRect
12344:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12345:  Function PolygonBounds( const P : array of TPoint) : TRect
12346:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12347:  Function RGBValue( const Color : TColor) : TRGB
12348:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12349:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12350:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12351:  Function TeeCull( const P : TFourPoints) : Boolean;
12352:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12353:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12354:  Procedure SmoothStretch( Src, Dst : TBitmap);
12355:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12356:  Function TeeDistance( const x, y : Double) : Double
12357:  Function TeeLoadLibrary( const FileName : String) : HInst
12358:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12359:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12360:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12361:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
      Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12362:   SIRegister_ICanvasHyperlinks(CL);
12363:   SIRegister_ICanvasToolTips(CL);
12364:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12365: end;
12366:
12367: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12368: begin
12369:   TOvcHdc', 'Integer
12370:   TOvcHWND', 'Cardinal
12371:   TOvcHdc', 'HDC
12372:   TOvcHWND', 'HWND
12373:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12374:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12375:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12376:  Function DefaultEpoch : Integer
12377:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12378:  Procedure FixRealPrim( P : PChar; DC : Char)
12379:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12380:  Function GetLeftButton : Byte
12381:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12382:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12383:  Function GetShiftFlags : Byte
12384:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12385:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12386:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12387:  Function ovIsForegroundTask : Boolean
```

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

```
12470:   SIRegister_TOvcAbstractStore(CL);
12471:   //PExPropInfo', '^TExPropInfo // will not work
12472: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12473:   SIRegister_TOvcPropertyList(CL);
12474:   SIRegister_TOvcDataFiler(CL);
12475:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12476:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12477:  Function CreateStoredItem( const CompName, PropName : string) : string
12478:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12479:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12480: end;
12481:
12482: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12483: begin
12484:  'ovsetsize','LongInt'( 16);
12485:  'etSyntax','LongInt'( 0);
12486:  'etSymantic','LongInt'( 1);
12487:  'chCR','Char #13);
12488:  'chLF','Char #10);
12489:  'chLineSeparator',' chCR);
12490:   SIRegister_TCocoError(CL);
12491:   SIRegister_TCommentItem(CL);
12492:   SIRegister_TCommentList(CL);
12493:   SIRegister_TSymbolPosition(CL);
12494: TGenListType', '( glNever, glAlways, glOnError )
12495: TovBitSet', 'set of Integer
12496:   //PStartTable', '^TStartTable // will not work
12497: 'TovCharSet', 'set of AnsiChar
12498: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12499: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12500: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12501: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12502: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12503:    +'osition; const Data : string; ErrorType : integer)
12504: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12505: TGetCH', 'Function ( pos : longint) : char
12506: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12507:   SIRegister_TCocoRScanner(CL);
12508:   SIRegister_TCocoRGrammar(CL);
12509:  '_EF','Char #0);
12510:  '_TAB','Char').SetString( #09);
12511:  '_CR','Char').SetString( #13);
12512:  '_LF','Char').SetString( #10);
12513:  '_EL','').SetString( _CR);
12514:  '_EOF','Char').SetString( #26);
12515:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12516:  'minErrDist','LongInt'( 2);
12517:  Function ovPadL( S : string; ch : char; L : integer) : string
12518: end;
12519:
12520:   TFormatSettings = record
12521:     CurrencyFormat: Byte;
12522:     NegCurrFormat: Byte;
12523:     ThousandSeparator: Char;
12524:     DecimalSeparator: Char;
12525:     CurrencyDecimals: Byte;
12526:     DateSeparator: Char;
12527:     TimeSeparator: Char;
12528:     ListSeparator: Char;
12529:     CurrencyString: string;
12530:     ShortDateFormat: string;
12531:     LongDateFormat: string;
12532:     TimeAMString: string;
12533:     TimePMString: string;
12534:     ShortTimeFormat: string;
12535:     LongTimeFormat: string;
12536:     ShortMonthNames: array[1..12] of string;
12537:     LongMonthNames: array[1..12] of string;
12538:     ShortDayNames: array[1..7] of string;
12539:     LongDayNames: array[1..7] of string;
12540:     TwoDigitYearCenturyWindow: Word;
12541:   end;
12542:
12543: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12544: begin
12545:  Function ovFormatSettings : TFormatSettings
12546: end;
12547:
12548: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12549: begin
12550:   TOvcCharSet', 'set of Char
12551:   ovBTable', 'array[0..255] of Byte
12552:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12553:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12554:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12555:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12556:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12557:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
```

```
12558:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12559:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12560:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12561:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12562:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12563:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12564:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12565:  Function LoCaseChar( C : Char) : Char
12566:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12567:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12568:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12569:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12570:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12571:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12572:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12573:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12574:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12575:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12576:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12577:  Procedure TrimAllSpacesPChar( P : PChar)
12578:  Function TrimEmbeddedZeros( const S : string) : string
12579:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12580:  Function TrimTrailPrimPChar( S : PChar) : PChar
12581:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12582:  Function TrimTrailingZeros( const S : string) : string
12583:  Procedure TrimTrailingZerosPChar( P : PChar)
12584:  Function UpCaseChar( C : Char) : Char
12585:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12586:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12587:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12588: end;
12589:
12590: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12591: begin
12592:   //PRaiseFrame', '^TRaiseFrame // will not work
12593:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12594:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12595:  Procedure SafeCloseHandle( var Handle : THandle)
12596:  Procedure ExchangeInteger( X1, X2 : Integer)
12597:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12598:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12599:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12600:
12601: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12602:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12603: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12604:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12605:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12606:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12607:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12608:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12609:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12610:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12611:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12612:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12613:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12614:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12615:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12616:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12617:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12618:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12619:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12620:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12621:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12622:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12623:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12624:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12625:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12626:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12627:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12628:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12629:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12630:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12631:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12632:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12633:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12634:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12635:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12636:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12637:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12638:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12639:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12640:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12641:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12642:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12643:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12644:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12645:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12646:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
```

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

```
12729:  Function
        CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12730:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
        nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12731:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
        lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
        Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
        lpProcessInfo:TProcessInformation):BOOL
12732:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
        Longint; lpName : PKOLChar) : THandle
12733:  Function
        wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12734:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12735:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12736:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12737:  //Function
        wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12738:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12739:  //Function
        wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12740:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12741:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12742:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12743:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12744:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12745:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12746:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
        dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12747:  Function wFindAtom( lpString : PKOLChar) : ATOM
12748:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12749:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12750:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12751:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12752:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12753:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12754:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12755:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12756:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12757:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12758:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12759:  Function wGetCommandLine : PKOLChar
12760:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12761:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12762:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12763:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12764:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12765:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12766:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12767:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12768:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12769:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12770:  Function wGetEnvironmentStrings : PKOLChar
12771:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12772:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12773:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12774:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12775:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12776:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12777:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12778:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12779:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12780:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12781:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12782:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12783:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12784:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12785:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12786:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12787:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12788:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12789:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12790: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12791:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
```

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

```
12860:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12861:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12862:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12863: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
       lpBufferSize:DWORD):DWORD;
12864:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12865: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12866: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12867:  //Function wWNetUseConnection(hwndOwner:HWND;var
       lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
       lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12868:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12869:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12870:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
       lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12871:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
       szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12872: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12873: //Func wGetPrivateProfileStruct(lpszSection,
       lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12874: //Func wWritePrivateProfileStruct(lpszSection,
       lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12875:  Function wAddFontResource( FileName : PKOLChar) : Integer
12876:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12877:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12878:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12879:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12880:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12881: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12882:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
       fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
       fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12883:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12884:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12885: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12886:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12887:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12888: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
       pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12889: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12890:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12891:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12892:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12893:  //Function wExtTextOut(DC:HDC;X,
       Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12894:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12895:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12896:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12897:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12898: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12899: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12900:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12901:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12902: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12903: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
       lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12904:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12905: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12906:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12907: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12908:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12909:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12910:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12911:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12912:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12913:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12914:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12915:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12916:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12917:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12918:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12919:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12920:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12921:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12922:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12923:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12924:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12925:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12926:  //Function
       wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12927:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12928: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
       dwFlags : DWORD; lParam : Pointer) : Longint
12929:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12930:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12931:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12932:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12933:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12934:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
```

```
12935: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12936:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12937:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12938:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12939:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12940:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12941:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12942:  //Function wCreateDesktop(lpszDesktop,
       lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12943:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
       HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12944:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
       lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12945:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
       hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12946: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
       nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12947:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
       dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12948:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12949:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12950:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12951:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12952:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
       : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12953:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
       : TFNDlgProc; dwInitParam : LPARAM) : Integer
12954:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12955:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
       nIDStaticPath:Integer;uFileType:UINT):Integer;
12956:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
       nIDStaticPath:Int;uFiletype:UINT):Int;
12957:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12958:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12959: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
       cy:Int;Flags:UINT):BOOL;
12960:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12961:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12962:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12963:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
       pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12964: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12965:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12966:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12967:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12968:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12969:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12970:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12971:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12972:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12973:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12974:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12975:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12976:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
       lpnTabStopPositions) : DWORD
12977: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
       lpnLengthNeed:DWORD)BOOL;
12978:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12979:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12980:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12981:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12982: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
       nHeigt:Int):BOOL;
12983:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12984:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12985:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12986:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12987:  Function wIsCharLower( ch : KOLChar) : BOOL
12988:  Function wIsCharUpper( ch : KOLChar) : BOOL
12989:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12990:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12991:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12992:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12993:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12994:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12995:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12996:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12997:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12998:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12999:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
13000:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
13001:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
13002:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
13003:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
13004:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
13005:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
13006:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
13007:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
13008:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
```

```
13009:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
13010:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
13011:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
13012:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
13013:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13014:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13015:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
13016: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
13017: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
13018:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
13019: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13020:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
13021:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
13022:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13023:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
13024:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
      lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
13025:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
      lpdwResult:DWORD): LRESULT
13026:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13027:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
13028:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
13029:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13030:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13031: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13032:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
13033:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13034:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
13035:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13036: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
13037:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
      lpnTabStopPositions,nTabOrigin:Int):Longint;
13038:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
13039:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13040:  Function wVkKeyScan( ch : KOLChar) : SHORT
13041:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13042:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
13043:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
13044:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
13045:
13046: //TestDrive!
13047: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13048:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13049:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
13050:  Function GetLocalUserSidStr( const UserName : string) : string
13051:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
13052:  Function Impersonate2User( const domain : string; const user : string) : boolean
13053:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
13054:  Function KillProcessbyname( const exename : string; var found : integer) : integer
13055:  Function getWinProcessList : TStringList
13056:  function WaitTilClose(hWnd: Integer): Integer;
13057:  function DoUserMsgs: Boolean;
13058:  function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13059:  procedure ShowMsg(hParent: Integer; const Mess, Title: String);  //modal but NOT blockable
13060:  procedure DeleteMsgForm(Handle: Integer);
13061:  procedure DisableForms;
13062:  function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13063: end;
13064:
13065: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13066: begin
13067:  'AfMaxSyncSlots','LongInt'( 64);
13068:  'AfSynchronizeTimeout','LongInt'( 2000);
13069:   TAfSyncSlotID', 'DWORD
13070:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13071:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13072:   TAfSafeDirectSyncEvent', 'Procedure
13073:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13074:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13075:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13076:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13077:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13078:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13079:  Function AfIsSyncMethod : Boolean
13080:  Function AfSyncWnd : HWnd
13081:  Function AfSyncStatistics : TAfSyncStatistics
13082:  Procedure AfClearSyncStatistics
13083: end;
13084:
13085: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13086: begin
13087:  'fBinary','LongWord')( $00000001);
13088:  'fParity','LongWord'( $00000002);
13089:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13090:  'fOutxDsrFlow','LongWord')( $00000008);
13091:  'fDtrControl','LongWord')( $00000030);
13092:  'fDtrControlDisable','LongWord')( $00000000);
13093:  'fDtrControlEnable','LongWord')( $00000010);
13094:  'fDtrControlHandshake','LongWord')( $00000020);
```

```
13095:  'fDsrSensitivity','LongWord')( $00000040);
13096:  'fTXContinueOnXoff','LongWord')( $00000080);
13097:  'fOutX','LongWord')( $00000100);
13098:  'fInX','LongWord')( $00000200);
13099:  'fErrorChar','LongWord')( $00000400);
13100:  'fNull','LongWord')( $00000800);
13101:  'fRtsControl','LongWord')( $00003000);
13102:  'fRtsControlDisable','LongWord')( $00000000);
13103:  'fRtsControlEnable','LongWord')( $00001000);
13104:  'fRtsControlHandshake','LongWord')( $00002000);
13105:  'fRtsControlToggle','LongWord')( $00003000);
13106:  'fAbortOnError','LongWord')( $00004000);
13107:  'fDummy2','LongWord')( $FFFF8000);
13108:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13109:   FindClass('TOBJECT'),'EAfComPortCoreError
13110:   FindClass('TOBJECT'),'TAfComPortCore
13111:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13112:    +'tKind : TAfCoreEvent; Data : DWORD)
13113:   SIRegister_TAfComPortCoreThread(CL);
13114:   SIRegister_TAfComPortEventThread(CL);
13115:   SIRegister_TAfComPortWriteThread(CL);
13116:   SIRegister_TAfComPortCore(CL);
13117:  Function FormatDeviceName( PortNumber : Integer) : string
13118: end;
13119:
13120: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13121: begin
13122:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13123:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13124:   SIRegister_TApplicationFileIO(CL);
13125:   TDataFileCapability', '( dfcRead, dfcWrite )
13126:   TDataFileCapabilities', 'set of TDataFileCapability
13127:   SIRegister_TDataFile(CL);
13128:   //TDataFileClass', 'class of TDataFile
13129:  Function ApplicationFileIODefined : Boolean
13130:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13131:  Function FileStreamExists(const fileName: String) : Boolean
13132:  //Procedure Register
13133: end;
13134:
13135: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13136: begin
13137:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13138:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13139:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13140:   TALFBXScale', 'Integer
13141:   FindClass('TOBJECT'),'EALFBXConvertError
13142:   SIRegister_EALFBXError(CL);
13143:   SIRegister_EALFBXException(CL);
13144:   FindClass('TOBJECT'),'EALFBXGFixError
13145:   FindClass('TOBJECT'),'EALFBXDSQLError
13146:   FindClass('TOBJECT'),'EALFBXDynError
13147:   FindClass('TOBJECT'),'EALFBXGBakError
13148:   FindClass('TOBJECT'),'EALFBXGSecError
13149:   FindClass('TOBJECT'),'EALFBXLicenseError
13150:   FindClass('TOBJECT'),'EALFBXGStatError
13151:   //EALFBXExceptionClass', 'class of EALFBXError
13152:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13153:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13154:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13155:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13156:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13157:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13158:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13159:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13160:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13161:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13162:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13163:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13164:   TALFBXTransParams', 'set of TALFBXTransParam
13165:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13166:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13167:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13168: 'cALFBXMaxParamLength','LongInt'( 125);
13169:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13170:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13171:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13172:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13173:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13174:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13175:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13176:   SIRegister_TALFBXSQLDA(CL);
13177:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13178:   SIRegister_TALFBXPoolStream(CL);
13179:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13180:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13181:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13182:   //TALFBXArrayDesc', 'TISCArrayDesc
13183:   //TALFBXBlobDesc', 'TISCBlobDesc
```

```
13184:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13185:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13186:   SIRegister_TALFBXSQLResult(CL);
13187:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13188:   SIRegister_TALFBXSQLParams(CL);
13189:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13190:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13191:    +'atementType : TALFBXStatementType; end
13192:   FindClass('TOBJECT'),'TALFBXLibrary
13193:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13194:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13195:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13196:    //+' Excep : EALFBXExceptionClass)
13197:   SIRegister_TALFBXLibrary(CL);
13198:  'cAlFBXDateOffset','LongInt'( 15018);
13199:  'cALFBXTimeCoeff','LongInt'( 864000000);
13200:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13201:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13202:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13203:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13204:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13205:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13206:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13207:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13208:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13209:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13210:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13211:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13212:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13213:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13214: end;
13215:
13216: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13217: begin
13218:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13219:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13220:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13221:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13222:    +'teger; First : Integer; CacheThreshold : Integer; end
13223:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13224:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13225:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13226:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13227:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13228:   SIRegister_TALFBXClient(CL);
13229:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13230:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13231:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13232:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13233:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13234:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13235:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13236:   SIRegister_TALFBXConnectionPoolClient(CL);
13237:   SIRegister_TALFBXEventThread(CL);
13238:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13239: end;
13240:
13241: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13242: begin
13243: _OSVERSIONINFOA = record
13244:     dwOSVersionInfoSize: DWORD;
13245:     dwMajorVersion: DWORD;
13246:     dwMinorVersion: DWORD;
13247:     dwBuildNumber: DWORD;
13248:     dwPlatformId: DWORD;
13249:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13250:   end;
13251:  TOSVersionInfoA', '_OSVERSIONINFOA
13252:   TOSVersionInfo', 'TOSVersionInfoA
13253:  'WS_EX_RIGHT','LongWord')( $00001000);
13254:  'WS_EX_LEFT','LongWord')( $00000000);
13255:  'WS_EX_RTLREADING','LongWord')( $00002000);
13256:  'WS_EX_LTRREADING','LongWord')( $00000000);
13257:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13258:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13259:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13260:  'LAYOUT_RTL','LongWord')( $00000001);
13261:  'LAYOUT_BTT','LongWord')( $00000002);
13262:  'LAYOUT_VBH','LongWord')( $00000004);
13263:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13264:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13265:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13266:  Function GetLayout( dc : hdc) : DWORD
13267:  Function IsBidi : Boolean
13268:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13269:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13270:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13271:  Function GetPriorityClass( hProcess : THandle) : DWORD
13272:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
```

```
13273:  Function CloseClipboard : BOOL
13274:  Function GetClipboardSequenceNumber : DWORD
13275:  Function GetClipboardOwner : HWND
13276:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13277:  Function GetClipboardViewer : HWND
13278:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13279:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13280:  Function GetClipboardData( uFormat : UINT) : THandle
13281:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13282:  Function CountClipboardFormats : Integer
13283:  Function EnumClipboardFormats( format : UINT) : UINT
13284:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13285:  Function EmptyClipboard : BOOL
13286:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13287:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13288:  Function GetOpenClipboardWindow : HWND
13289:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13290:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13291:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13292:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13293:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13294:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13295:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13296:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13297:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13298: end;
13299:
13300: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13301: begin
13302:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13303:  Function GetTemporaryFilesPath : String
13304:  Function GetTemporaryFileName : String
13305:  Function FindFileInPaths( const fileName, paths : String) : String
13306:  Function PathsToString( const paths : TStrings) : String
13307:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13308:  //Function MacroExpandPath( const aPath : String) : String
13309: end;
13310:
13311: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13312: begin
13313:   SIRegister_TALMultiPartBaseContent(CL);
13314:   SIRegister_TALMultiPartBaseContents(CL);
13315:   SIRegister_TAlMultiPartBaseStream(CL);
13316:   SIRegister_TALMultipartBaseEncoder(CL);
13317:   SIRegister_TALMultipartBaseDecoder(CL);
13318:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13319:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13320:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13321: end;
13322:
13323: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13324: begin
13325:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13326:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13327:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13328:  Function aAllocPadedMem( Size : Cardinal) : TObject
13329:  Procedure aFreePadedMem( var P : TObject);
13330:  Procedure aFreePadedMem1( var P : PChar);
13331:  Function aCheckPadedMem( P : Pointer) : Byte
13332:  Function aGetPadMemSize( P : Pointer) : Cardinal
13333:  Function aAllocMem( Size : Cardinal) : Pointer
13334:  Function aStrLen( const Str : PChar) : Cardinal
13335:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13336:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13337:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13338:  Function aStrEnd( const Str : PChar) : PChar
13339:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13340:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13341:  Function aPCharLength( const Str : PChar) : Cardinal
13342:  Function aPCharUpper( Str : PChar) : PChar
13343:  Function aPCharLower( Str : PChar) : PChar
13344:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13345:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13346:  Function aCopyTail( const S : String; Len : Integer) : String
13347:  Function aInt2Thos( I : Int64) : String
13348:  Function aUpperCase( const S : String) : String
13349:  Function aLowerCase( const S : string) : String
13350:  Function aCompareText( const S1, S2 : string) : Integer
13351:  Function aSameText( const S1, S2 : string) : Boolean
13352:  Function aInt2Str( Value : Int64) : String
13353:  Function aStr2Int( const Value : String) : Int64
13354:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13355:  Function aGetFileExt( const FileName : String) : String
13356:  Function aGetFilePath( const FileName : String) : String
13357:  Function aGetFileName( const FileName : String) : String
13358:  Function aChangeExt( const FileName, Extension : String) : String
13359:  Function aAdjustLineBreaks( const S : string) : string
13360:  Function aGetWindowStr( WinHandle : HWND) : String
13361:  Function aDiskSpace( Drive : String) : TdriveSize
```

```
13362:  Function aFileExists( FileName : String) : Boolean
13363:  Function aFileSize( FileName : String) : Int64
13364:  Function aDirectoryExists( const Name : string) : Boolean
13365:  Function aSysErrorMessage( ErrorCode : Integer) : string
13366:  Function aShortPathName( const LongName : string) : string
13367:  Function aGetWindowVer : TWinVerRec
13368:  procedure InitDriveSpacePtr;
13369: end;
13370:
13371: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13372: begin
13373:  aZero','LongInt'( 0);
13374:  'makeappDEF','LongInt'( - 1);
13375:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13376:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13377:  'CS_KEYCVTWINDOW','LongInt'( 4);
13378:  'CS_DBLCLKS','LongInt'( 8);
13379:  'CS_OWNDC','LongWord')( $20);
13380:  'CS_CLASSDC','LongWord')( $40);
13381:  'CS_PARENTDC','LongWord')( $80);
13382:  'CS_NOKEYCVT','LongWord')( $100);
13383:  'CS_NOCLOSE','LongWord')( $200);
13384:  'CS_SAVEBITS','LongWord')( $800);
13385:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13386:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13387:  'CS_GLOBALCLASS','LongWord')( $4000);
13388:  'CS_IME','LongWord')( $10000);
13389:  'CS_DROPSHADOW','LongWord')( $20000);
13390:   //PPanelFunc', '^TPanelFunc // will not work
13391:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13392:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13393:   TFontLooks', 'set of TFontLook
13394:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13395:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13396:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13397:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13398:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13399:  Procedure RunMsgLoop( Show : Boolean)
13400:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13401:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13402:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13403:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13404:  Function MakePanel(Left,Top,Width,Height,
       hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13405:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13406:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13407:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13408: end;
13409:
13410: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13411: begin
13412:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13413:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13414:   TScreenSaverOptions', 'set of TScreenSaverOption
13415:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13416:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13417:   SIRegister_TScreenSaver(CL);
13418:  //Procedure Register
13419:  Procedure SetScreenSaverPassword
13420: end;
13421:
13422: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13423: begin
13424:   FindClass('TOBJECT'),'TXCollection
13425:   SIRegister_EFilerException(CL);
13426:   SIRegister_TXCollectionItem(CL);
13427:   //TXCollectionItemClass', 'class of TXCollectionItem
13428:   SIRegister_TXCollection(CL);
13429:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13430:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13431:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13432:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13433:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13434:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13435: end;
13436:
13437: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13438: begin
13439:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13440:  Procedure xglMapTexCoordToNull
13441:  Procedure xglMapTexCoordToMain
13442:  Procedure xglMapTexCoordToSecond
13443:  Procedure xglMapTexCoordToDual
13444:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13445:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13446:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13447:  Procedure xglBeginUpdate
```

```
13448:  Procedure xglEndUpdate
13449:  Procedure xglPushState
13450:  Procedure xglPopState
13451:  Procedure xglForbidSecondTextureUnit
13452:  Procedure xglAllowSecondTextureUnit
13453:  Function xglGetBitWiseMapping : Cardinal
13454: end;
13455:
13456: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13457: begin
13458:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13459:   TBaseListOptions', 'set of TBaseListOption
13460:   SIRegister_TBaseList(CL);
13461:   SIRegister_TBaseVectorList(CL);
13462:   SIRegister_TAffineVectorList(CL);
13463:   SIRegister_TVectorList(CL);
13464:   SIRegister_TTexPointList(CL);
13465:   SIRegister_TXIntegerList(CL);
13466:   //PSingleArrayList', '^TSingleArrayList // will not work
13467:   SIRegister_TSingleList(CL);
13468:   SIRegister_TByteList(CL);
13469:   SIRegister_TQuaternionList(CL);
13470:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13471:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13472:  Procedure FastQuickSortLists(startIndex,
      endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13473: end;
13474:
13475: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13476: begin
13477:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13478:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13479:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
      indices:TIntegerList;list:TAffineVectorList);
13480:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
      indices:TIntegerList;list:TAffineVectorList);
13481:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
      normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13482:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13483:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13484:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13485:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13486:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13487:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13488:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13489:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13490:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13491:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
      edgesTriangles : TIntegerList) : TIntegerList
13492:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13493:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
      TPersistentObjectList;
13494:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13495:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
      TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13496: end;
13497:
13498: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13499: begin
13500:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13501:  Procedure FreeMemAndNil( var P : TObject)
13502:  Function PCharOrNil( const S : string) : PChar
13503:   SIRegister_TJclReferenceMemoryStream(CL);
13504:   FindClass('TOBJECT'),'EJclVMTError
13505:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13506:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13507:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13508:   PDynamicIndexList', '^TDynamicIndexList // will not work
13509:   PDynamicAddressList', '^TDynamicAddressList // will not work
13510:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13511:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13512:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13513:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13514:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13515:  Function GetInitTable( AClass : TClass) : PTypeInfo
13516:   PFieldEntry', '^TFieldEntry // will not work}
13517:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13518:  Function JIsClass( Address : Pointer) : Boolean
13519:  Function JIsObject( Address : Pointer) : Boolean
13520:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13521:   TDigitCount', 'Integer
13522:   SIRegister_TJclNumericFormat(CL);
13523:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13524:   TTextHandler', 'Procedure ( const Text : string)
13525: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13526:  Function JExecute(const
      CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13527:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13528:  Function ReadKey : Char   //to and from the DOS console !
```

```
13529:   TModuleHandle', 'HINST
13530:   //TModuleHandle', 'Pointer
13531:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13532:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13533:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13534:  Procedure UnloadModule( var Module : TModuleHandle)
13535:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13536:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13537:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13538:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13539:   FindClass('TOBJECT'),'EJclConversionError
13540:  Function JStrToBoolean( const S : string) : Boolean
13541:  Function JBooleanToStr( B : Boolean) : string
13542:  Function JIntToBool( I : Integer) : Boolean
13543:  Function JBoolToInt( B : Boolean) : Integer
13544:  'ListSeparator','String ';
13545:  'ListSeparator1','String ':
13546:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13547:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13548:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13549:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13550:  Function ListItemCount( const List, Separator : string) : Integer
13551:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13552:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13553:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13554:  Function SystemTObjectInstance : LongWord
13555:  Function IsCompiledWithPackages : Boolean
13556:  Function JJclGUIDToString( const GUID : TGUID) : string
13557:  Function JJclStringToGUID( const S : string) : TGUID
13558:   SIRegister_TJclIntfCriticalSection(CL);
13559:   SIRegister_TJclSimpleLog(CL);
13560:  Procedure InitSimpleLog( const ALogFileName : string)
13561: end;
13562:
13563: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13564: begin
13565:   FindClass('TOBJECT'),'EJclBorRADException
13566:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13567:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13568:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13569:   TJclBorRADToolPath', 'string
13570:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13571:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13572:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13573:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13574:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13575:  BorRADToolRepositoryFormsPage','String 'Forms
13576:  BorRADToolRepositoryProjectsPage','String 'Projects
13577:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13578:  BorRADToolRepositoryObjectType','String 'Type
13579:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13580:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13581:  BorRADToolRepositoryObjectName','String 'Name
13582:  BorRADToolRepositoryObjectPage','String 'Page
13583:  BorRADToolRepositoryObjectIcon','String 'Icon
13584:  BorRADToolRepositoryObjectDescr','String 'Description
13585:  BorRADToolRepositoryObjectAuthor','String 'Author
13586:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13587:  BorRADToolRepositoryObjectDesigner','String 'Designer
13588:  BorRADToolRepositoryDesignerDfm','String 'dfm
13589:  BorRADToolRepositoryDesignerXfm','String 'xfm
13590:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13591:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13592:  SourceExtensionDelphiPackage','String '.dpk
13593:  SourceExtensionBCBPackage','String '.bpk
13594:  SourceExtensionDelphiProject','String '.dpr
13595:  SourceExtensionBCBProject','String '.bpr
13596:  SourceExtensionBDSProject','String '.bdsproj
13597:  SourceExtensionDProject','String '.dproj
13598:  BinaryExtensionPackage','String '.bpl
13599:  BinaryExtensionLibrary','String '.dll
13600:  BinaryExtensionExecutable','String '.exe
13601:  CompilerExtensionDCP','String '.dcp
13602:  CompilerExtensionBPI','String '.bpi
13603:  CompilerExtensionLIB','String '.lib
13604:  CompilerExtensionTDS','String '.tds
13605:  CompilerExtensionMAP','String '.map
13606:  CompilerExtensionDRC','String '.drc
13607:  CompilerExtensionDEF','String '.def
13608:  SourceExtensionCPP','String '.cpp
13609:  SourceExtensionH','String '.h
13610:  SourceExtensionPAS','String '.pas
13611:  SourceExtensionDFM','String '.dfm
13612:  SourceExtensionXFM','String '.xfm
13613:  SourceDescriptionPAS','String 'Pascal source file
13614:  SourceDescriptionCPP','String 'C++ source file
13615:  DesignerVCL','String 'VCL
13616:  DesignerCLX','String 'CLX
13617:  ProjectTypePackage','String 'package
```

```
13618:  ProjectTypeLibrary','String 'library
13619:  ProjectTypeProgram','String 'program
13620:  Personality32Bit','String '32 bit
13621:  Personality64Bit','String '64 bit
13622:  PersonalityDelphi','String 'Delphi
13623:  PersonalityDelphiDotNet','String 'Delphi.net
13624:  PersonalityBCB','String 'C++Builder
13625:  PersonalityCSB','String 'C#Builder
13626:  PersonalityVB','String 'Visual Basic
13627:  PersonalityDesign','String 'Design
13628:  PersonalityUnknown','String 'Unknown personality
13629:  PersonalityBDS','String 'Borland Developer Studio
13630:  DOFDirectoriesSection','String 'Directories
13631:  DOFUnitOutputDirKey','String 'UnitOutputDir
13632:  DOFSearchPathName','String 'SearchPath
13633:  DOFConditionals','String 'Conditionals
13634:  DOFLinkerSection','String 'Linker
13635:  DOFPackagesKey','String 'Packages
13636:  DOFCompilerSection','String 'Compiler
13637:  DOFPackageNoLinkKey','String 'PackageNoLink
13638:  DOFAdditionalSection','String 'Additional
13639:  DOFOptionsKey','String 'Options
13640:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13641:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13642:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13643:   TJclBorPersonalities', 'set of TJclBorPersonality
13644:   TJclBorDesigner', '( bdVCL, bdCLX )
13645:   TJclBorDesigners', 'set of TJClBorDesigner
13646:   TJclBorPlatform', '( bp32bit, bp64bit )
13647:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13648:   SIRegister_TJclBorRADToolInstallationObject(CL);
13649:   SIRegister_TJclBorlandOpenHelp(CL);
13650:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13651:   TJclHelp2Objects', 'set of TJclHelp2Object
13652:   SIRegister_TJclHelp2Manager(CL);
13653:   SIRegister_TJclBorRADToolIdeTool(CL);
13654:   SIRegister_TJclBorRADToolIdePackages(CL);
13655:   SIRegister_IJclCommandLineTool(CL);
13656:   FindClass('TOBJECT'),'EJclCommandLineToolError
13657:   SIRegister_TJclCommandLineTool(CL);
13658:   SIRegister_TJclBorlandCommandLineTool(CL);
13659:   SIRegister_TJclBCC32(CL);
13660:   SIRegister_TJclDCC32(CL);
13661:   TJclDCC', 'TJclDCC32
13662:   SIRegister_TJclBpr2Mak(CL);
13663:   SIRegister_TJclBorlandMake(CL);
13664:   SIRegister_TJclBorRADToolPalette(CL);
13665:   SIRegister_TJclBorRADToolRepository(CL);
13666:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13667:   TCommandLineTools', 'set of TCommandLineTool
13668:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13669:   SIRegister_TJclBorRADToolInstallation(CL);
13670:   SIRegister_TJclBCBInstallation(CL);
13671:   SIRegister_TJclDelphiInstallation(CL);
13672:   SIRegister_TJclDCCIL(CL);
13673:   SIRegister_TJclBDSInstallation(CL);
13674:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13675:   SIRegister_TJclBorRADToolInstallations(CL);
13676:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13677:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13678:  Function IsDelphiPackage( const FileName : string) : Boolean
13679:  Function IsDelphiProject( const FileName : string) : Boolean
13680:  Function IsBCBPackage( const FileName : string) : Boolean
13681:  Function IsBCBProject( const FileName : string) : Boolean
13682:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13683:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13684:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13685:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
13686:  function SamePath(const Path1, Path2: string): Boolean;
13687: end;
13688:
13689: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13690: begin
13691:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13692:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13693:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13694:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13695:  'LPathSeparator','String '/
13696:  'LDirDelimiter','String '/
13697:  'LDirSeparator','String ':
13698:  'JXPathDevicePrefix','String '\\.\
13699:  'JXPathSeparator','String '\
13700:  'JXDirDelimiter','String '\
13701:  'JXDirSeparator','String ';
13702:  'JXPathUncPrefix','String '\\
13703:  'faNormalFile','LongWord')( $00000080);
13704:  //'faUnixSpecific',' faSymLink);
```

```
13705:   JXTCompactPath', '( cpCenter, cpEnd )
13706:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13707:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13708:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13709:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13710:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13711:
13712:   Function jxPathAddSeparator( const Path : string) : string
13713:   Function jxPathAddExtension( const Path, Extension : string) : string
13714:   Function jxPathAppend( const Path, Append : string) : string
13715:   Function jxPathBuildRoot( const Drive : Byte) : string
13716:   Function jxPathCanonicalize( const Path : string) : string
13717:   Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13718:   Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13719:   Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13720:   Function jxPathExtractFileDirFixed( const S : string) : string
13721:   Function jxPathExtractFileNameNoExt( const Path : string) : string
13722:   Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13723:   Function jxPathGetDepth( const Path : string) : Integer
13724:   Function jxPathGetLongName( const Path : string) : string
13725:   Function jxPathGetShortName( const Path : string) : string
13726:   Function jxPathGetLongName( const Path : string) : string
13727:   Function jxPathGetShortName( const Path : string) : string
13728:   Function jxPathGetRelativePath( Origin, Destination : string) : string
13729:   Function jxPathGetTempPath : string
13730:   Function jxPathIsAbsolute( const Path : string) : Boolean
13731:   Function jxPathIsChild( const Path, Base : string) : Boolean
13732:   Function jxPathIsDiskDevice( const Path : string) : Boolean
13733:   Function jxPathIsUNC( const Path : string) : Boolean
13734:   Function jxPathRemoveSeparator( const Path : string) : string
13735:   Function jxPathRemoveExtension( const Path : string) : string
13736:   Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13737:   Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13738:    JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13739:    JxTFileListOptions', 'set of TFileListOption
13740:    JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13741:    TFileHandler', 'Procedure ( const FileName : string)
13742:    TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13743:   Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13744:   //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
      AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
      FileMatchFunc:TFileMatchFunc):Bool;
13745:   Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13746:   Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13747:   Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13748:   Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13749:   Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
      RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13750:   Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
      IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13751:   Procedure jxCreateEmptyFile( const FileName : string)
13752:   Function jxCloseVolume( var Volume : THandle) : Boolean
13753:   Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13754:   Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13755:   Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13756:   Function jxDelTree( const Path : string) : Boolean
13757:   //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13758:   Function jxDiskInDrive( Drive : Char) : Boolean
13759:   Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13760:   Function jxFileCreateTemp( var Prefix : string) : THandle
13761:   Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13762:   Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13763:   Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13764:   Function jxFileExists( const FileName : string) : Boolean
13765:   Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13766:   Function jxFileRestore( const FileName : string) : Boolean
13767:   Function jxGetBackupFileName( const FileName : string) : string
13768:   Function jxIsBackupFileName( const FileName : string) : Boolean
13769:   Function jxFileGetDisplayName( const FileName : string) : string
13770:   Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13771:   Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13772:   Function jxFileGetSize( const FileName : string) : Int64
13773:   Function jxFileGetTempName( const Prefix : string) : string
13774:   Function jxFileGetTypeName( const FileName : string) : string
13775:   Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13776:   Function jxForceDirectories( Name : string) : Boolean
13777:   Function jxGetDirectorySize( const Path : string) : Int64
13778:   Function jxGetDriveTypeStr( const Drive : Char) : string
13779:   Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13780:   Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13781:   Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13782:   Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13783:   Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13784:   //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
      ResolveSymLinks:Boolean):Integer
13785:   Function jxGetFileLastWrite( const FName : string) : TFileTime;
13786:   Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13787:   Function jxGetFileLastAccess( const FName : string) : TFileTime;
13788:   Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
```

```
13789:  Function jxGetFileCreation( const FName : string) : TFileTime;
13790:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13791:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13792:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13793:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13794:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13795:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13796:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13797:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13798:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13799:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13800:  Function jxGetModulePath( const Module : HMODULE) : string
13801:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13802:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13803:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13804:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13805:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13806:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13807:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13808:  Function jxOpenVolume( const Drive : Char) : THandle
13809:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13810:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13811:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13812:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13813:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13814:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13815:  Procedure jxShredFile( const FileName : string; Times : Integer)
13816:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13817:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13818:  Function jxSymbolicLinkTarget( const Name : string) : string
13819:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13820:   SIRegister_TJclCustomFileAttrMask(CL);
13821:   SIRegister_TJclFileAttributeMask(CL);
13822:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13823:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13824:   TFileSearchOptions', 'set of TFileSearchOption
13825:   TFileSearchTaskID', 'Integer
13826:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13827:    +'hTaskID; const Aborted : Boolean)
13828:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13829:   SIRegister_IJclFileEnumerator(CL);
13830:   SIRegister_TJclFileEnumerator(CL);
13831:  Function JxFileSearch : IJclFileEnumerator
13832:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13833:   JxTFileFlags', 'set of TFileFlag
13834:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13835:   SIRegister_TJclFileVersionInfo(CL);
13836:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13837:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13838:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13839:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13840:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13841:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13842:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13843:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13844:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13845:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13846:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13847:   SIRegister_TJclTempFileStream(CL);
13848:   FindClass('TOBJECT'),'TJclCustomFileMapping
13849:   SIRegister_TJclFileMappingView(CL);
13850:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13851:   SIRegister_TJclCustomFileMapping(CL);
13852:   SIRegister_TJclFileMapping(CL);
13853:   SIRegister_TJclSwapFileMapping(CL);
13854:   SIRegister_TJclFileMappingStream(CL);
13855:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13856:   //PPCharArray', '^TPCharArray // will not work
13857:   SIRegister_TJclMappedTextReader(CL);
13858:   SIRegister_TJclFileMaskComparator(CL);
13859:   FindClass('TOBJECT'),'EJclPathError
13860:   FindClass('TOBJECT'),'EJclFileUtilsError
13861:   FindClass('TOBJECT'),'EJclTempFileStreamError
13862:   FindClass('TOBJECT'),'EJclTempFileStreamError
13863:   FindClass('TOBJECT'),'EJclFileMappingError
13864:   FindClass('TOBJECT'),'EJclFileMappingViewError
13865:  Function jxPathGetLongName2( const Path : string) : string
13866:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13867:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13868:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13869:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13870:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13871:  Procedure jxPathListAddItems( var List : string; const Items : string)
13872:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13873:  Procedure jxPathListDelItems( var List : string; const Items : string)
13874:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13875:  Function jxPathListItemCount( const List : string) : Integer
```

```
13876:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13877:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13878:  Function jxPathListItemIndex( const List, Item : string) : Integer
13879:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13880:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13881:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13882:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string) : Integer
13883: end;
13884:
13885: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13886: begin
13887:  'UTF8FileHeader','String #$ef#$bb#$bf);
13888:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13889:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13890:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13891:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13892:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13893:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13894:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13895:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13896:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13897:  Function lFileIsReadable( const AFilename : string) : boolean
13898:  Function lFileIsWritable( const AFilename : string) : boolean
13899:  Function lFileIsText( const AFilename : string) : boolean
13900:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13901:  Function lFileIsExecutable( const AFilename : string) : boolean
13902:  Function lFileIsSymlink( const AFilename : string) : boolean
13903:  Function lFileIsHardLink( const AFilename : string) : boolean
13904:  Function lFileSize( const Filename : string) : int64;
13905:  Function lGetFileDescription( const AFilename : string) : string
13906:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13907:  Function lTryReadAllLinks( const Filename : string) : string
13908:  Function lDirPathExists( const FileName : String) : Boolean
13909:  Function lForceDirectory( DirectoryName : string) : boolean
13910:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13911:  Function lProgramDirectory : string
13912:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13913:  Function lExtractFileNameOnly( const AFilename : string) : string
13914:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13915:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13916:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13917:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13918:  Function lAppendPathDelim( const Path : string) : string
13919:  Function lChompPathDelim( const Path : string) : string
13920:  Function lTrimFilename( const AFilename : string) : string
13921:  Function lCleanAndExpandFilename( const Filename : string) : string
13922:  Function lCleanAndExpandDirectory( const Filename : string) : string
13923:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13924:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13925:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13926:  Function lFileIsInPath( const Filename, Path : string) : boolean
13927:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13928:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13929:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13930:  'AllDirectoryEntriesMask','String '*
13931:  Function lGetAllFilesMask : string
13932:  Function lGetExeExt : string
13933:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13934:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13935:  Function lFindDiskFilename( const Filename : string) : string
13936:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13937:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13938:  Function lGetDarwinSystemFilename( Filename : string) : string
13939:   SIRegister_TFileIterator(CL);
13940:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13941:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13942:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13943:   SIRegister_TFileSearcher(CL);
13944:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13945:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13946:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13947:  // TCopyFileFlags', 'set of TCopyFileFlag
13948:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13949:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13950:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13951:  Function lReadFileToString( const Filename : string) : string
13952:  Function lGetTempFilename( const Directory, Prefix : string) : string
13953:  {Function NeedRTLAnsi : boolean
13954:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13955:  Function UTF8ToSys( const s : string) : string
13956:  Function SysToUTF8( const s : string) : string
13957:  Function ConsoleToUTF8( const s : string) : string
13958:  Function UTF8ToConsole( const s : string) : string}
13959:  Function FileExistsUTF8( const Filename : string) : boolean
```

```
13960:  Function FileAgeUTF8( const FileName : string) : Longint
13961:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13962:  Function ExpandFileNameUTF8( const FileName : string) : string
13963:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13964:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13965:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13966:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13967:  Procedure FindCloseUTF8( var F : TSearchrec)
13968:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13969:  Function FileGetAttrUTF8( const FileName : String) : Longint
13970:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13971:  Function DeleteFileUTF8( const FileName : String) : Boolean
13972:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13973:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13974:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13975:  Function GetCurrentDirUTF8 : String
13976:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13977:  Function CreateDirUTF8( const NewDir : String) : Boolean
13978:  Function RemoveDirUTF8( const Dir : String) : Boolean
13979:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13980:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13981:  Function FileCreateUTF8( const FileName : string) : THandle;
13982:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13983:  Function ParamStrUTF8( Param : Integer) : string
13984:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13985:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13986:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13987:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13988:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13989: end;
13990:
13991: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13992: begin
13993:   //VK_F23 = 134;
13994:   //{$EXTERNALSYM VK_F24}
13995:   //VK_F24 = 135;
13996:  TVirtualKeyCode', 'Integer
13997:  'VK_MOUSEWHEELUP','integer'(134);
13998:  'VK_MOUSEWHEELDOWN','integer'(135);
13999:  Function glIsKeyDown( c : Char) : Boolean;
14000:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
14001:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
14002:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
14003:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
14004:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
14005:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
14006: end;
14007:
14008: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14009: begin
14010:   TGLPoint', 'TPoint
14011:   //PGLPoint', '^TGLPoint // will not work
14012:   TGLRect', 'TRect
14013:   //PGLRect', '^TGLRect // will not work
14014:   TDelphiColor', 'TColor
14015:   TGLPicture', 'TPicture
14016:   TGLGraphic', 'TGraphic
14017:   TGLBitmap', 'TBitmap
14018:   //TGraphicClass', 'class of TGraphic
14019:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14020:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14021:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14022:    +'Button; Shift : TShiftState; X, Y : Integer)
14023:   TGLMouseMoveEvent', 'TMouseMoveEvent
14024:   TGLKeyEvent', 'TKeyEvent
14025:   TGLKeyPressEvent', 'TKeyPressEvent
14026:   EGLOSError', 'EWin32Error
14027:   EGLOSError', 'EWin32Error
14028:   EGLOSError', 'EOSError
14029:  'glsAllFilter','string'All // sAllFilter
14030:  Function GLPoint( const x, y : Integer) : TGLPoint
14031:  Function GLRGB( const r, g, b : Byte) : TColor
14032:  Function GLColorToRGB( color : TColor) : TColor
14033:  Function GLGetRValue( rgb : DWORD) : Byte
14034:  Function GLGetGValue( rgb : DWORD) : Byte
14035:  Function GLGetBValue( rgb : DWORD) : Byte
14036:  Procedure GLInitWinColors
14037:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
14038:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
14039:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14040:  Procedure GLInformationDlg( const msg : String)
14041:  Function GLQuestionDlg( const msg : String) : Boolean
14042:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14043:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14044:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14045:  Function GLApplicationTerminated : Boolean
14046:  Procedure GLRaiseLastOSError
14047:  Procedure GLFreeAndNil( var anObject: TObject)
14048:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
```

```
14049:  Function GLGetCurrentColorDepth : Integer
14050:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14051:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14052:  Procedure GLSleep( length : Cardinal)
14053:  Procedure GLQueryPerformanceCounter( var val : Int64)
14054:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14055:  Function GLStartPrecisionTimer : Int64
14056:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14057:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14058:  Function GLRDTSC : Int64
14059:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14060:  Function GLOKMessageBox( const Text, Caption : string) : Integer
14061:  Procedure GLShowHTMLUrl( Url : String)
14062:  Procedure GLShowCursor( AShow : boolean)
14063:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14064:  Procedure GLGetCursorPos( var point : TGLPoint)
14065:  Function GLGetScreenWidth : integer
14066:  Function GLGetScreenHeight : integer
14067:  Function GLGetTickCount : int64
14068:  function RemoveSpaces(const str : String) : String;
14069:   TNormalMapSpace','( nmsObject, nmsTangent )
14070:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14071:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14072:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14073:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
14074:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14075: end;
14076:
14077: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14078: begin
14079:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14080:  // PGLStarRecord', '^TGLStarRecord // will not work
14081:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14082:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14083:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14084: end;
14085:
14086:
14087: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14088: begin
14089:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14090:   //PAABB', '^TAABB // will not work
14091:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14092:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14093:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14094:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14095:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14096:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14097:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14098:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14099:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14100:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14101:  Function BBMinX( const c : THmgBoundingBox) : Single
14102:  Function BBMaxX( const c : THmgBoundingBox) : Single
14103:  Function BBMinY( const c : THmgBoundingBox) : Single
14104:  Function BBMaxY( const c : THmgBoundingBox) : Single
14105:  Function BBMinZ( const c : THmgBoundingBox) : Single
14106:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14107:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14108:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14109:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14110:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14111:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14112:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14113:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14114:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14115:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14116:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14117:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14118:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14119:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14120:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14121:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14122:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14123:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14124:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14125:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14126:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14127:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14128:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14129:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14130:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14131:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14132:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14133:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
```

```
14134:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14135:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14136:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14137:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14138:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14139:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
      viewportSizeY:Int):TClipRect
14140: end;
14141:
14142: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14143: begin
14144:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14145:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14146:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14147:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14148:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14149:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14150:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14151:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14152:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14153:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14154:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14155:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14156:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14157:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14158:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14159:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14160:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14161:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14162:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14163:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14164:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14165:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14166:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14167:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14168:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14169: end;
14170:
14171: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14172: begin
14173:  'EPSILON','Single').setExtended( 1e-40);
14174:  'EPSILON2','Single').setExtended( 1e-30);  }
14175:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14176:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14177:  THmgPlane', 'TVector
14178:   TDoubleHmgPlane', 'THomogeneousDblVector
14179:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14180:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14181:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14182:   TSingleArray', 'array of Single
14183:   TTransformations','array [0..15] of Single)
14184:   TPackedRotationMatrix','array [0..2] of Smallint)
14185:   TVertex', 'TAffineVector
14186:   //TVectorGL', 'THomogeneousFltVector
14187:   //TMatrixGL', 'THomogeneousFltMatrix
14188:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14189:  Function glTexPointMake( const s, t : Single) : TTexPoint
14190:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14191:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14192:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14193:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14194:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14195:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14196:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14197:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14198:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14199:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14200:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14201:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14202:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14203:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14204:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14205:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14206:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14207:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14208:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14209:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14210:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14211:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14212:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14213:  Procedure glRstVector( var v : TAffineVector);
14214:  Procedure glRstVector1( var v : TVectorGL);
14215:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14216:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14217:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14218:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14219:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14220:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14221:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
```

```
14222:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14223:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14224:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14225:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14226:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14227:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
       nb:Int;dest:PTexPointArray);
14228:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
       nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14229:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
       PAffineVectorArray);
14230:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14231:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14232:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14233:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14234:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14235:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14236:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14237:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14238:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14239:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14240:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14241:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14242:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14243:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14244:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14245:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14246:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14247:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14248:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14249:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14250:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14251:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14252:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14253:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14254:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14255:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14256:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14257:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14258:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14259:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14260:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14261:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14262:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14263:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14264:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14265:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14266:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14267:  Function glLerp( const start, stop, t : Single) : Single
14268:  Function glAngleLerp( start, stop, t : Single) : Single
14269:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14270:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14271:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14272:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14273:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14274:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14275:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14276:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14277: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14278: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
       PAffineVectorArray);
14279:  Function glVectorLength( const x, y : Single) : Single;
14280:  Function glVectorLength1( const x, y, z : Single) : Single;
14281:  Function glVectorLength2( const v : TAffineVector) : Single;
14282:  Function glVectorLength3( const v : TVectorGL) : Single;
14283:  Function glVectorLength4( const v : array of Single) : Single;
14284:  Function glVectorNorm( const x, y : Single) : Single;
14285:  Function glVectorNorm1( const v : TAffineVector) : Single;
14286:  Function glVectorNorm2( const v : TVectorGL) : Single;
14287:  Function glVectorNorm3( var V : array of Single) : Single;
14288:  Procedure glNormalizeVector( var v : TAffineVector);
14289:  Procedure glNormalizeVector1( var v : TVectorGL);
14290:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14291:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14292: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14293:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14294:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14295:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14296:  Procedure glNegateVector( var V : TAffineVector);
14297:  Procedure glNegateVector2( var V : TVectorGL);
14298:  Procedure glNegateVector3( var V : array of Single);
14299:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14300:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14301:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14302:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14303:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14304:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14305:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14306:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
```

```
14307:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14308:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14309:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14310:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14311:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14312:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14313:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14314:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14315:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14316:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14317:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14318:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14319:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14320:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14321:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14322:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14323:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14324:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14325:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14326:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14327:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14328:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14329:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14330:  Procedure glAbsVector( var v : TVectorGL);
14331:  Procedure glAbsVector1( var v : TAffineVector);
14332:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14333:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14334:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14335:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14336:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14337:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14338:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14339:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14340:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14341:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14342:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14343:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14344:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14345:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14346:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14347:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14348:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14349:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14350:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14351:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14352:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14353:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14354:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14355:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14356:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14357:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14358:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14359:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14360:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14361:  Procedure glAdjointMatrix( var M : TMatrixGL);
14362:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14363:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14364:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14365:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14366:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14367:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14368:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14369:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14370:  Procedure glInvertMatrix( var M : TMatrixGL);
14371:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14372:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14373:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14374:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14375:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14376:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14377:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14378:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14379:  Procedure glNormalizePlane( var plane : THmgPlane)
14380:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14381:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14382:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14383:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14384:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14385:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14386:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14387:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14388:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14389:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14390:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14391:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14392:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14393:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14394:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
```

```
14395:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14396:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14397:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14398:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14399:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14400:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14401:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14402:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14403:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14404:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14405:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14406:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14407:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14408:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14409:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14410:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14411:  Function glLnXP1( X : Extended) : Extended
14412:  Function glLog10( X : Extended) : Extended
14413:  Function glLog2( X : Extended) : Extended;
14414:  Function glLog21( X : Single) : Single;
14415:  Function glLogN( Base, X : Extended) : Extended
14416:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14417:  Function glPower( const Base, Exponent : Single) : Single;
14418:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14419:  Function glDegToRad( const Degrees : Extended) : Extended;
14420:  Function glDegToRad1( const Degrees : Single) : Single;
14421:  Function glRadToDeg( const Radians : Extended) : Extended;
14422:  Function glRadToDeg1( const Radians : Single) : Single;
14423:  Function glNormalizeAngle( angle : Single) : Single
14424:  Function glNormalizeDegAngle( angle : Single) : Single
14425:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14426:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14427:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14428:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14429:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14430:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14431:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14432:  Function glArcCos( const X : Extended) : Extended;
14433:  Function glArcCos1( const x : Single) : Single;
14434:  Function glArcSin( const X : Extended) : Extended;
14435:  Function glArcSin1( const X : Single) : Single;
14436:  Function glArcTan21( const Y, X : Extended) : Extended;
14437:  Function glArcTan21( const Y, X : Single) : Single;
14438:  Function glFastArcTan2( y, x : Single) : Single
14439:  Function glTan( const X : Extended) : Extended;
14440:  Function glTan1( const X : Single) : Single;
14441:  Function glCoTan( const X : Extended) : Extended;
14442:  Function glCoTan1( const X : Single) : Single;
14443:  Function glSinh( const x : Single) : Single;
14444:  Function glSinh1( const x : Double) : Double;
14445:  Function glCosh( const x : Single) : Single;
14446:  Function glCosh1( const x : Double) : Double;
14447:  Function glRSqrt( v : Single) : Single
14448:  Function glRLength( x, y : Single) : Single
14449:  Function glISqrt( i : Integer) : Integer
14450:  Function glILength( x, y : Integer) : Integer;
14451:  Function glILength1( x, y, z : Integer) : Integer;
14452:  Procedure glRegisterBasedExp
14453:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14454:  Function glRoundInt( v : Single) : Single;
14455:  Function glRoundInt1( v : Extended) : Extended;
14456:  Function glTrunc( v : Single) : Integer;
14457:  Function glTrunc64( v : Extended) : Int64;
14458:  Function glInt( v : Single) : Single;
14459:  Function glInt1( v : Extended) : Extended;
14460:  Function glFrac( v : Single) : Single;
14461:  Function glFrac1( v : Extended) : Extended;
14462:  Function glRound( v : Single) : Integer;
14463:  Function glRound64( v : Single) : Int64;
14464:  Function glRound641( v : Extended) : Int64;
14465:  Function glTrunc( X : Extended) : Int64
14466:  Function glRound( X : Extended) : Int64
14467:  Function glFrac( X : Extended) : Extended
14468:  Function glCeil( v : Single) : Integer;
14469:  Function glCeil64( v : Extended) : Int64;
14470:  Function glFloor( v : Single) : Integer;
14471:  Function glFloor64( v : Extended) : Int64;
14472:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14473:  Function glSign( x : Single) : Integer
14474:  Function glIsInRange( const x, a, b : Single) : Boolean;
14475:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14476:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14477:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14478:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14479:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14480:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14481:  Function glMinFloat3( const v1, v2 : Single) : Single;
14482:  Function glMinFloat4( const v : array of Single) : Single;
14483:  Function glMinFloat5( const v1, v2 : Double) : Double;
```

```
14484:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14485:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14486:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14487:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14488:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14489:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14490:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14491:  Function glMaxFloat2( const v : array of Single) : Single;
14492:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14493:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14494:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14495:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14496:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14497:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14498:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14499:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14500:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14501:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14502:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14503:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14504:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14505:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14506:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14507:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14508:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14509:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14510:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14511:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14512:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14513:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14514:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14515:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14516:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14517:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14518:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14519:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14520:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14521:  Procedure glSortArrayAscending( var a : array of Extended)
14522:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14523:  Function glClampValue1( const aValue, aMin : Single) : Single;
14524:  Function glGeometryOptimizationMode : String
14525:  Procedure glBeginFPUOnlySection
14526:  Procedure glEndFPUOnlySection
14527:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14528:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14529:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14530:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14531:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14532:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14533:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14534:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14535:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14536:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14537:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14538:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14539:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14540:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14541:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14542:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14543:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14544:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14545:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14546:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14547:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14548:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14549:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14550:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14551:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14552:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14553:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14554:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14555:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14556:  'cPI','Single').setExtended( 3.141592654);
14557:  'cPIdiv180','Single').setExtended( 0.017453292);
14558:  'c180divPI','Single').setExtended( 57.29577951);
14559:  'c2PI','Single').setExtended( 6.283185307);
14560:  'cPIdiv2','Single').setExtended( 1.570796326);
14561:  'cPIdiv4','Single').setExtended( 0.785398163);
14562:  'c3PIdiv4','Single').setExtended( 2.35619449);
14563:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14564:  'cInv360','Single').setExtended( 1 / 360);
14565:  'c180','Single').setExtended( 180);
14566:  'c360','Single').setExtended( 360);
14567:  'cOneHalf','Single').setExtended( 0.5);
```

```
14568:  'cLn10','Single').setExtended( 2.302585093);
14569:  {'MinSingle','Extended').setExtended( 1.5e-45);
14570:  'MaxSingle','Extended').setExtended( 3.4e+38);
14571:  'MinDouble','Extended').setExtended( 5.0e-324);
14572:  'MaxDouble','Extended').setExtended( 1.7e+308);
14573:  'MinExtended','Extended').setExtended( 3.4e-4932);
14574:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14575:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14576:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14577: end;
14578:
14579: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14580: begin
14581:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14582:   (FindClass('TOBJECT'),'TFaceGroups
14583:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14584:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14585:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14586:   SIRegister_TBaseMeshObject(CL);
14587:   (FindClass('TOBJECT'),'TSkeletonFrameList
14588:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14589:   SIRegister_TSkeletonFrame(CL);
14590:   SIRegister_TSkeletonFrameList(CL);
14591:   (FindClass('TOBJECT'),'TSkeleton
14592:   (FindClass('TOBJECT'),'TSkeletonBone
14593:   SIRegister_TSkeletonBoneList(CL);
14594:   SIRegister_TSkeletonRootBoneList(CL);
14595:   SIRegister_TSkeletonBone(CL);
14596:   (FindClass('TOBJECT'),'TSkeletonColliderList
14597:   SIRegister_TSkeletonCollider(CL);
14598:   SIRegister_TSkeletonColliderList(CL);
14599:   (FindClass('TOBJECT'),'TGLBaseMesh
14600:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14601:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14602:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14603:    +'QuaternionList; end
14604:   SIRegister_TSkeleton(CL);
14605:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14606:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14607:   SIRegister_TMeshObject(CL);
14608:   SIRegister_TMeshObjectList(CL);
14609:   //TMeshObjectListClass', 'class of TMeshObjectList
14610:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14611:   SIRegister_TMeshMorphTarget(CL);
14612:   SIRegister_TMeshMorphTargetList(CL);
14613:   SIRegister_TMorphableMeshObject(CL);
14614:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14615:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14616:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14617:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14618:   SIRegister_TSkeletonMeshObject(CL);
14619:   SIRegister_TFaceGroup(CL);
14620:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14621:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14622:   SIRegister_TFGVertexIndexList(CL);
14623:   SIRegister_TFGVertexNormalTexIndexList(CL);
14624:   SIRegister_TFGIndexTexCoordList(CL);
14625:   SIRegister_TFaceGroups(CL);
14626:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14627:   SIRegister_TVectorFile(CL);
14628:   //TVectorFileClass', 'class of TVectorFile
14629:   SIRegister_TGLGLSMVectorFile(CL);
14630:   SIRegister_TGLBaseMesh(CL);
14631:   SIRegister_TGLFreeForm(CL);
14632:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14633:   TGLActorOptions', 'set of TGLActorOption
14634:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14635:   (FindClass('TOBJECT'),'TGLActor
14636:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14637:   SIRegister_TActorAnimation(CL);
14638:   TActorAnimationName', 'String
14639:   SIRegister_TActorAnimations(CL);
14640:   SIRegister_TGLBaseAnimationControler(CL);
14641:   SIRegister_TGLAnimationControler(CL);
14642:   TActorFrameInterpolation', '( afpNone, afpLinear )
14643:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14644:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14645:   SIRegister_TGLActor(CL);
14646:   SIRegister_TVectorFileFormat(CL);
14647:   SIRegister_TVectorFileFormatsList(CL);
14648:   (FindClass('TOBJECT'),'EInvalidVectorFile
14649:  Function GetVectorFileFormats : TVectorFileFormatsList
14650:  Function VectorFileFormatsFilter : String
14651:  Function VectorFileFormatsSaveFilter : String
14652:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14653:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14654:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14655: end;
14656:
```

```
14657: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14658: begin
14659:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14660:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14661:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14662:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14663:   SIRegister_TOleStream(CL);
14664:   (FindClass('TOBJECT'),'TConnectionPoints
14665:   TConnectionKind', '( ckSingle, ckMulti )
14666:   SIRegister_TConnectionPoint(CL);
14667:   SIRegister_TConnectionPoints(CL);
14668:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14669:   (FindClass('TOBJECT'),'TActiveXControlFactory
14670:   SIRegister_TActiveXControl(CL);
14671:   //TActiveXControlClass', 'class of TActiveXControl
14672:   SIRegister_TActiveXControlFactory(CL);
14673:   SIRegister_TActiveFormControl(CL);
14674:   SIRegister_TActiveForm(CL);
14675:   //TActiveFormClass', 'class of TActiveForm
14676:   SIRegister_TActiveFormFactory(CL);
14677:   (FindClass('TOBJECT'),'TPropertyPageImpl
14678:   SIRegister_TPropertyPage(CL);
14679:   //TPropertyPageClass', 'class of TPropertyPage
14680:   SIRegister_TPropertyPageImpl(CL);
14681:   SIRegister_TActiveXPropertyPage(CL);
14682:   SIRegister_TActiveXPropertyPageFactory(CL);
14683:   SIRegister_TCustomAdapter(CL);
14684:   SIRegister_TAdapterNotifier(CL);
14685:   SIRegister_IFontAccess(CL);
14686:   SIRegister_TFontAdapter(CL);
14687:   SIRegister_IPictureAccess(CL);
14688:   SIRegister_TPictureAdapter(CL);
14689:   SIRegister_TOleGraphic(CL);
14690:   SIRegister_TStringsAdapter(CL);
14691:   SIRegister_TReflectorWindow(CL);
14692:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14693:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14694:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14695:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14696:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14697:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14698:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14699:  Function ParkingWindow : HWND
14700: end;
14701:
14702: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14703: begin
14704:  // TIp6Bytes = array [0..15] of Byte;
14705: {:binary form of IPv6 adress (for string conversion routines)}
14706:  // TIp6Words = array [0..7] of Word;
14707:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14708:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14709:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14710:  Function synaIsIP( const Value : string) : Boolean');
14711:  Function synaIsIP6( const Value : string) : Boolean');
14712:  Function synaIPToID( Host : string) : Ansistring');
14713:  Function synaStrToIp6( value : string) : TIp6Bytes');
14714:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14715:  Function synaStrToIp( value : string) : integer');
14716:  Function synaIpToStr( value : integer) : string');
14717:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14718:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14719:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14720:  Function xStrToIP( const Value : String) : TIPAdr');
14721:  Function xIPToStr( const Adresse : TIPAdr) : String');
14722:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14723:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14724: end;
14725:
14726: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14727: begin
14728:  AddTypeS('TSpecials', 'set of Char');
14729:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14730:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14731:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14732:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14733:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14734:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14735:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14736:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14737:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14738:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14739:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14740:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14741:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14742:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14743:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14744:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14745:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
```

```
14746:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14747:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14748:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14749:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14750:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14751:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14752:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14753:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14754:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14755:  Function synCrc32( const Value : AnsiString) : Integer');
14756:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14757:  Function Crc16( const Value : AnsiString) : Word');
14758:  Function synMD5( const Value : AnsiString) : AnsiString');
14759:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14760:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14761:  Function synSHA1( const Value : AnsiString) : AnsiString');
14762:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14763:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14764:  Function synMD4( const Value : AnsiString) : AnsiString');
14765: end;
14766:
14767: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14768: begin
14769:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14770:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14771:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14772:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14773:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14774:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14775:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14776:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14777:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14778:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14779:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14780:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14781:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14782:    +', CP864, CP865, CP869, CP1125 )');
14783:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14784:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14785:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
      TransformTable : array of Word) : AnsiString');
14786:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
      TransformTable : array of Word; Translit : Boolean) : AnsiString');
14787:  Function GetCurCP : TMimeChar');
14788:  Function GetCurOEMCP : TMimeChar');
14789:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14790:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14791:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14792:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14793:  Function GetBOM( Value : TMimeChar) : AnsiString');
14794:  Function StringToWide( const Value : AnsiString) : WideString');
14795:  Function WideToString( const Value : WideString) : AnsiString');
14796: end;
14797:
14798: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14799: begin
14800:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14801:  Procedure WakeOnLan( MAC, IP : string)');
14802:  Function GetDNS : string');
14803:  Function GetIEProxy( protocol : string) : TProxySetting');
14804:  Function GetLocalIPs : string');
14805: end;
14806:
14807:
14808: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14809: begin
14810:  AddConstantN('synCR','Char #$0d);
14811:  Const('synLF','Char #$0a);
14812:  Const('cSerialChunk','LongInt'( 8192);
14813:  Const('LockfileDirectory','String '/var/lock');
14814:  Const('PortIsClosed','LongInt'( - 1);
14815:  Const('ErrAlreadyOwned','LongInt'( 9991);
14816:  Const('ErrAlreadyInUse','LongInt'( 9992);
14817:  Const('ErrWrongParameter','LongInt'( 9993);
14818:  Const('ErrPortNotOpen','LongInt'( 9994);
14819:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14820:  Const('ErrMaxBuffer','LongInt'( 9996);
14821:  Const('ErrTimeout','LongInt'( 9997);
14822:  Const('ErrNotRead','LongInt'( 9998);
14823:  Const('ErrFrame','LongInt'( 9999);
14824:  Const('ErrOverrun','LongInt'( 10000);
14825:  Const('ErrRxOver','LongInt'( 10001);
14826:  Const('ErrRxParity','LongInt'( 10002);
14827:  Const('ErrTxFull','LongInt'( 10003);
14828:  Const('dcb_Binary','LongWord')( $00000001);
14829:  Const('dcb_ParityCheck','LongWord')( $00000002);
14830:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14831:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14832:  Const('dcb_DtrControlMask','LongWord')( $00000030);
```

```
14833:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14834:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14835:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14836:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14837:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14838:  Const('dcb_OutX','LongWord')( $00000100);
14839:  Const('dcb_InX','LongWord')( $00000200);
14840:  Const('dcb_ErrorChar','LongWord')( $00000400);
14841:  Const('dcb_NullStrip','LongWord')( $00000800);
14842:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14843:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14844:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14845:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14846:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14847:  Const('dcb_AbortOnError','LongWord')( $00004000);
14848:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14849:  Const('synSB1','LongInt'( 0);
14850:  Const('SB1andHalf','LongInt'( 1);
14851:  Const('synSB2','LongInt'( 2);
14852:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14853:  Const('CS7fix','LongWord')( $0000020);
14854:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14855:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14856:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14857:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14858:   //AddTypeS('PDCB', '^TDCB // will not work');
14859:  //Const('MaxRates','LongInt'( 18);
14860:  //Const('MaxRates','LongInt'( 30);
14861:  //Const('MaxRates','LongInt'( 19);
14862:  Const('O_SYNC','LongWord')( $0080);
14863:  Const('synOK','LongInt'( 0);
14864:  Const('synErr','LongInt'( integer ( - 1 ));
14865:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14866:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14867:   SIRegister_ESynaSerError(CL);
14868:   SIRegister_TBlockSerial(CL);
14869:  Function GetSerialPortNames : string');
14870: end;
14871:
14872: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14873: begin
14874:  Const('DLLIconvName','String 'libiconv.so');
14875:  Const('DLLIconvName','String 'iconv.dll');
14876:   AddTypeS('size_t', 'Cardinal');
14877:   AddTypeS('iconv_t', 'Integer');
14878:   //AddTypeS('iconv_t', 'Pointer');
14879:   AddTypeS('argptr', 'iconv_t');
14880:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14881:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14882:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14883:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14884:  Function SynaIconvClose( var cd : iconv_t) : integer');
14885:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14886:  Function IsIconvloaded : Boolean');
14887:  Function InitIconvInterface : Boolean');
14888:  Function DestroyIconvInterface : Boolean');
14889:  Const('ICONV_TRIVIALP','LongInt'( 0);
14890:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14891:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14892:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14893:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14894: end;
14895:
14896: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14897: begin
14898:  Const 'ICMP_ECHO','LongInt'( 8);
14899:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14900:  Const('ICMP_UNREACH','LongInt'( 3);
14901:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14902:  Const('ICMP6_ECHO','LongInt'( 128);
14903:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14904:  Const('ICMP6_UNREACH','LongInt'( 1);
14905:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14906:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14907:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14908:   SIRegister_TPINGSend(CL);
14909:  Function PingHost( const Host : string) : Integer');
14910:  Function TraceRouteHost( const Host : string) : string');
14911: end;
14912:
14913: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14914: begin
14915:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14916:  Const('synASN1_INT','LongWord')( $02);
14917:  Const('synASN1_OCTSTR','LongWord')( $04);
14918:  Const('synASN1_NULL','LongWord')( $05);
14919:  Const('synASN1_OBJID','LongWord')( $06);
14920:  Const('synASN1_ENUM','LongWord')( $0a);
```

```
14921:  Const('synASN1_SEQ','LongWord')( $30);
14922:  Const('synASN1_SETOF','LongWord')( $31);
14923:  Const('synASN1_IPADDR','LongWord')( $40);
14924:  Const('synASN1_COUNTER','LongWord')( $41);
14925:  Const('synASN1_GAUGE','LongWord')( $42);
14926:  Const('synASN1_TIMETICKS','LongWord')( $43);
14927:  Const('synASN1_OPAQUE','LongWord')( $44);
14928:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14929:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14930:  Function synASNEncLen( Len : Integer) : AnsiString');
14931:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14932:  Function synASNEncInt( Value : Integer) : AnsiString');
14933:  Function synASNEncUInt( Value : Integer) : AnsiString');
14934:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14935:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14936:  Function synMibToId( Mib : String) : AnsiString');
14937:  Function synIdToMib( const Id : AnsiString) : String');
14938:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14939:  Function ASNdump( const Value : AnsiString) : AnsiString');
14940:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14941:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14942: end;
14943:
14944: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14945: begin
14946:  Const('cLDAPProtocol','String '389');
14947:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14948:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14949:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14950:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14951:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14952:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14953:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14954:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14955:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14956:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14957:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14958:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14959:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14960:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14961:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14962:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14963:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14964:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14965:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14966:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14967:   SIRegister_TLDAPAttribute(CL);
14968:   SIRegister_TLDAPAttributeList(CL);
14969:   SIRegister_TLDAPResult(CL);
14970:   SIRegister_TLDAPResultList(CL);
14971:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14972:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14973:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14974:   SIRegister_TLDAPSend(CL);
14975:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14976: end;
14977:
14978:
14979: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14980: begin
14981:  Const('cSysLogProtocol','String '514');
14982:  Const('FCL_Kernel','LongInt'( 0);
14983:  Const('FCL_UserLevel','LongInt'( 1);
14984:  Const('FCL_MailSystem','LongInt'( 2);
14985:  Const('FCL_System','LongInt'( 3);
14986:  Const('FCL_Security','LongInt'( 4);
14987:  Const('FCL_Syslogd','LongInt'( 5);
14988:  Const('FCL_Printer','LongInt'( 6);
14989:  Const('FCL_News','LongInt'( 7);
14990:  Const('FCL_UUCP','LongInt'( 8);
14991:  Const('FCL_Clock','LongInt'( 9);
14992:  Const('FCL_Authorization','LongInt'( 10);
14993:  Const('FCL_FTP','LongInt'( 11);
14994:  Const('FCL_NTP','LongInt'( 12);
14995:  Const('FCL_LogAudit','LongInt'( 13);
14996:  Const('FCL_LogAlert','LongInt'( 14);
14997:  Const('FCL_Time','LongInt'( 15);
14998:  Const('FCL_Local0','LongInt'( 16);
14999:  Const('FCL_Local1','LongInt'( 17);
15000:  Const('FCL_Local2','LongInt'( 18);
15001:  Const('FCL_Local3','LongInt'( 19);
15002:  Const('FCL_Local4','LongInt'( 20);
15003:  Const('FCL_Local5','LongInt'( 21);
15004:  Const('FCL_Local6','LongInt'( 22);
15005:  Const('FCL_Local7','LongInt'( 23);
15006:   Type(TSyslogSeverity','(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15007:   SIRegister_TSyslogMessage(CL);
15008:   SIRegister_TSyslogSend(CL);
15009:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
       Content:string):Boolean;
```

```
15010: end;
15011:
15012:
15013: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
15014: begin
15015:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15016:   SIRegister_TMessHeader(CL);
15017:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
15018:   SIRegister_TMimeMess(CL);
15019: end;
15020:
15021: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
15022: begin
15023:   (FindClass('TOBJECT'),'TMimePart');
15024:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15025:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15026:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15027:   SIRegister_TMimePart(CL);
15028:  Const('MaxMimeType','LongInt'( 25);
15029:  Function GenerateBoundary : string');
15030: end;
15031:
15032: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15033: begin
15034:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
15035:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
15036:  Function NeedInline( const Value : AnsiString) : boolean');
15037:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15038:  Function InlineCode( const Value : string) : string');
15039:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15040:  Function InlineEmail( const Value : string) : string');
15041: end;
15042:
15043: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15044: begin
15045:  Const('cFtpProtocol','String '21');
15046:  Const('cFtpDataProtocol','String '20');
15047:  Const('FTP_OK','LongInt'( 255);
15048:  Const('FTP_ERR','LongInt'( 254);
15049:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15050:   SIRegister_TFTPListRec(CL);
15051:   SIRegister_TFTPList(CL);
15052:   SIRegister_TFTPSend(CL);
15053:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15054:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15055:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
       ToPort, ToFile, ToUser, ToPass : string) : Boolean');
15056: end;
15057:
15058: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15059: begin
15060:  Const('cHttpProtocol','String '80');
15061:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15062:   SIRegister_THTTPSend(CL);
15063:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
15064:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
15065:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
15066:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
15067:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
       ResultData:TStrings):Bool;
15068: end;
15069:
15070: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15071: begin
15072:  Const('cSmtpProtocol','String '25');
15073:   SIRegister_TSMTPSend(CL);
15074:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
       Passw:string):Bool;
15075:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15076:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
       Username, Password : string):Boolean');
15077: end;
15078:
15079: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15080: begin
15081:  Const('cSnmpProtocol','String '161');
15082:  Const('cSnmpTrapProtocol','String '162');
15083:  Const('SNMP_V1','LongInt'( 0);
15084:  Const('SNMP_V2C','LongInt'( 1);
15085:  Const('SNMP_V3','LongInt'( 3);
15086:  Const('PDUGetRequest','LongWord')( $A0);
15087:  Const('PDUGetNextRequest','LongWord')( $A1);
15088:  Const('PDUGetResponse','LongWord')( $A2);
15089:  Const('PDUSetRequest','LongWord')( $A3);
15090:  Const('PDUTrap','LongWord')( $A4);
15091:  Const('PDUGetBulkRequest','LongWord')( $A5);
15092:  Const('PDUInformRequest','LongWord')( $A6);
15093:  Const('PDUTrapV2','LongWord')( $A7);
15094:  Const('PDUReport','LongWord')( $A8);
```

```
15095:  Const('ENoError',LongInt 0;
15096:  Const('ETooBig','LongInt')( 1);
15097:  Const('ENoSuchName','LongInt'( 2);
15098:  Const('EBadValue','LongInt'( 3);
15099:  Const('EReadOnly','LongInt'( 4);
15100:  Const('EGenErr','LongInt'( 5);
15101:  Const('ENoAccess','LongInt'( 6);
15102:  Const('EWrongType','LongInt'( 7);
15103:  Const('EWrongLength','LongInt'( 8);
15104:  Const('EWrongEncoding','LongInt'( 9);
15105:  Const('EWrongValue','LongInt'( 10);
15106:  Const('ENoCreation','LongInt'( 11);
15107:  Const('EInconsistentValue','LongInt'( 12);
15108:  Const('EResourceUnavailable','LongInt'( 13);
15109:  Const('ECommitFailed','LongInt'( 14);
15110:  Const('EUndoFailed','LongInt'( 15);
15111:  Const('EAuthorizationError','LongInt'( 16);
15112:  Const('ENotWritable','LongInt'( 17);
15113:  Const('EInconsistentName','LongInt'( 18);
15114:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15115:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15116:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15117:   SIRegister_TSNMPMib(CL);
15118:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15119:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15120:   SIRegister_TSNMPRec(CL);
15121:   SIRegister_TSNMPSend(CL);
15122:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15123:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15124:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15125:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15126:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15127:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15128:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : TStringList) : Integer');
15129: end;
15130:
15131: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15132: begin
15133:  Function GetDomainName2: AnsiString');
15134:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15135:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15136:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15137:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15138:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15139: end;
15140:
15141: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15142: begin
15143:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15144:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15145:  Function wwStrToDate( const S : string) : boolean');
15146:  Function wwStrToTime( const S : string) : boolean');
15147:  Function wwStrToDateTime( const S : string) : boolean');
15148:  Function wwStrToTimeVal( const S : string) : TDateTime');
15149:  Function wwStrToDateVal( const S : string) : TDateTime');
15150:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15151:  Function wwStrToInt( const S : string) : boolean');
15152:  Function wwStrToFloat( const S : string) : boolean');
15153:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15154:  Function wwNextDay( Year, Month, Day : Word) : integer');
15155:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15156:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15157:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15158:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15159:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15160:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15161:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15162:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15163:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15164: end;
15165:
15166: unit uPSI_Themes;
15167: Function ThemeServices : TThemeServices');
15168: Function ThemeControl( AControl : TControl) : Boolean');
15169: Function UnthemedDesigner( AControl : TControl) : Boolean');
15170: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15171: begin
15172:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15173: end;
15174: Unit uPSC_menus;
15175:  Function StripHotkey( const Text : string) : string');
15176:  Function GetHotkey( const Text : string) : string');
15177:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15178:  Function IsAltGRPressed : boolean');
15179:
15180: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15181: begin
```

```
15182:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15183:   SIRegister_TIdIMAP4Server(CL);
15184: end;
15185:
15186: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15187: begin
15188:  'HASH_SIZE','LongInt'( 256);
15189:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15190:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15191:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15192:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15193:    +' : Integer; Value : Variant; end');
15194:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15195:   SIRegister_TVariantSymbolTable(CL);
15196: end;
15197:
15198: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15199: begin
15200:   SIRegister_TThreadLocalVariables(CL);
15201:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15202:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15203:  Function ThreadLocals : TThreadLocalVariables');
15204:  Procedure WriteDebug( sz : String)');
15205:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15206:  'UDF_FAILURE','LongInt'( 1);
15207:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15208:  CL.AddTypeS('mTByteArray', 'array of byte;');
15209:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15210:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15211:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15212:  function IsNetworkConnected: Boolean;
15213:  function IsInternetConnected: Boolean;
15214:  function IsCOMConnected: Boolean;
15215:  function IsNetworkOn: Boolean;
15216:  function IsInternetOn: Boolean;
15217:  function IsCOMOn: Boolean;
15218:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15219:  TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15220:  Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT');
15221:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15222:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15223:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15224:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15225:  Function GetMenu( hWnd : HWND) : HMENU');
15226:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15227: end;
15228:
15229: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15230: begin
15231:   SIRegister_IDataBlock(CL);
15232:   SIRegister_ISendDataBlock(CL);
15233:   SIRegister_ITransport(CL);
15234:   SIRegister_TDataBlock(CL);
15235:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15236:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15237:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15238:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15239:   SIRegister_TCustomDataBlockInterpreter(CL);
15240:   SIRegister_TSendDataBlock(CL);
15241:   'CallSig','LongWord')( $D800);
15242:   'ResultSig','LongWord')( $D400);
15243:   'asMask','LongWord')( $00FF);
15244:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15245:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15246:  Procedure CheckSignature( Sig : Integer)');
15247: end;
15248:
15249: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15250: begin
15251:   //CL.AddTypeS('HINTERNET', '___Pointer');
15252:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15253:   CL.AddTypeS('HINTERNET', 'Integer');
15254:   CL.AddTypeS('HINTERNET2', '___Pointer');
15255:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15256:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15257:   CL.AddTypeS('INTERNET_PORT', 'Word');
15258:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15259:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15260:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15261:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15262:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15263:  Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
      lpUrlComponents:TURLComponents):BOOL;
15264:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
      dwUrlLength:DWORD):BOOL;
15265:  Function InternetCloseHandle(hInet : HINTERNET) : BOOL');
15266:  Function
      InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
      lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
```

```
15267:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15268:  ;dwContext:DWORD):HINTERNET;
15269:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
        lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15270:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
        dwContext:DWORD):BOOL;
15271:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15272:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15273:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15274:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15275:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15276:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15277:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15278:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
        lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15279:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
        lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15280:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15281:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15282:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
        dwContext:DWORD):BOOL;
15283:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
        TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15284:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
        BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15285:  Function
        WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15286:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15287:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15288:  Function
        FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15289:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15290:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15291:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15292:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15293:  Function
        FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15294:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15295:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15296:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15297:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15298:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15299:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15300:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15301:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15302:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15303:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15304:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15305:  Function
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
        PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15306:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15307:  Function
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15308:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
        PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15309:  Function
        HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15310:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
        dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15311:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15312:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15313:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15314:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15315:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15316:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15317:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15318:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15319:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
        lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15320:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15321:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15322:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15323:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15324: end;
15325:
15326: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15327: begin
15328:   AddTypeS('strCharSet', 'set of char');
15329:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15330:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15331:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15332:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15333:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15334:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15335:  Procedure strStripWhiteSpace( var s : string)');
```

```
15336:  Function strRemoveChar( str : string; removeChar : char) : string');
15337:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15338:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15339:  Function wwEqualStr( s1, s2 : string) : boolean');
15340:  Function strCount( s : string; delimeter : char) : integer');
15341:  Function strWhiteSpace : strCharSet');
15342:  Function wwExtractFileNameOnly( const FileName : string) : string');
15343:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15344:  Function strTrailing( s : string; delimeter : char) : string');
15345:  Function strPreceding( s : string; delimeter : char) : string');
15346:  Function wwstrReplace( s, Find, Replace : string) : string');
15347: end;
15348:
15349: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15350: begin
15351:   SIRegister_TRemoteDataModule(CL);
15352:   SIRegister_TCRemoteDataModule(CL);
15353:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15354:  Procedure UnregisterPooled( const ClassID : string)');
15355:  Procedure EnableSocketTransport( const ClassID : string)');
15356:  Procedure DisableSocketTransport( const ClassID : string)');
15357:  Procedure EnableWebTransport( const ClassID : string)');
15358:  Procedure DisableWebTransport( const ClassID : string)');
15359: end;
15360:
15361: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15362: begin
15363:  Function mxArcCos( x : Real) : Real');
15364:  Function mxArcSin( x : Real) : Real');
15365:  Function Comp2Str( N : Comp) : String');
15366:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15367:  Function Int2Str( N : LongInt) : String');
15368:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15369:  Function LogXY( x, y : Real) : Real');
15370:  Function Pennies2Dollars( C : Comp) : String');
15371:  Function mxPower( X : Integer; Y : Integer) : Real');
15372:  Function Real2Str( N : Real; Width, Places : integer) : String');
15373:  Function mxStr2Comp( MyString : string) : Comp');
15374:  Function mxStr2Pennies( S : String) : Comp');
15375:  Function Str2Real( MyString : string) : Real');
15376:  Function XToTheY( x, y : Real) : Real');
15377: end;
15378:
15379: //************************Cindy Functions!*****************************
15380: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15381: begin
15382:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15383:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15384:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15385:  MessagePlainText','String 'text/plain');
15386:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15387:  MessageAlterText_Html','String 'multipart/alternative');
15388:  MessageHtml_Attach','String 'multipart/mixed');
15389:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15390:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15391:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15392:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15393:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15394:  Function ForceDecodeHeader( aHeader : String) : String');
15395:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15396:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15397:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15398:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15399:  Function Get_MD5( const aFileName : string) : string');
15400:  Function Get_MD5FromString( const aString : string) : string');
15401: end;
15402:
15403: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15404: begin
15405:  Function IsFolder( SRec : TSearchrec) : Boolean');
15406:  Function isFolderReadOnly( Directory : String) : Boolean');
15407:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15408:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15409:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15410:  Function DiskFreeBytes( Drv : Char) : Int64');
15411:  Function DiskBytes( Drv : Char) : Int64');
15412:  Function GetFileBytes( Filename : String) : Int64');
15413:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15414:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15415:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15416:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15417:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15418:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15419:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15420:  SE_TCB_NAME','String 'SeTcbPrivilege');
15421:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15422:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15423:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15424:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
```

```
15425:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15426:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15427:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15428:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15429:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15430:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15431:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15432:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15433:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15434:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15435:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15436:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15437:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15438:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15439:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15440:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15441:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15442: end;
15443:
15444: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15445: begin
15446:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15447:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15448:  Function ShellGetExtensionName( FileName : String) : String');
15449:  Function ShellGetIconIndex( FileName : String) : Integer');
15450:  Function ShellGetIconHandle( FileName : String) : HIcon');
15451:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15452:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15453:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15454:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15455:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15456:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15457:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15458:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
       WaitCloseCompletion : boolean) : Boolean');
15459:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15460:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15461:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15462:  Function GetModificationDate( Filename : String) : TDateTime');
15463:  Function GetCreationDate( Filename : String) : TDateTime');
15464:  Function GetLastAccessDate( Filename : String) : TDateTime');
15465:  Function FileDelete( Filename : String) : Boolean');
15466:  Function FileIsOpen( Filename : string) : boolean');
15467:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15468:  Function DirectoryDelete( Directory : String) : Boolean');
15469:  Function GetPrinters( PrintersList : TStrings) : Integer');
15470:  Procedure SetDefaultPrinter( PrinterName : String)');
15471:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15472:  Function WinToDosPath( WinPathName : String) : String');
15473:  Function DosToWinPath( DosPathName : String) : String');
15474:  Function cyGetWindowsVersion : TWindowsVersion');
15475:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15476:  Procedure WindowsShutDown( Restart : boolean)');
15477:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15478:  Procedure GetWindowsFonts( FontsList : TStrings)');
15479:  Function GetAvailableFilename( DesiredFileName : String): String');
15480: end;
15481:
15482: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15483: begin
15484:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15485:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15486:   Type(TStringRead', '( srFromLeft, srFromRight )');
15487:   Type(TStringReads', 'set of TStringRead');
15488:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15489:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15490:   Type(TWordsOptions', 'set of TWordsOption');
15491:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15492:   Type(TCarTypes', 'set of TCarType');
15493:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15494:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15495:  Function Char_GetType( aChar : Char) : TCarType');
15496:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15497:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15498:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15499:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15500:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15501:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15502:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15503:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15504:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15505:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15506:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15507:  Function String_Quote( Str : String) : String');
15508:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15509:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15510:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15511:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
```

```
15512:  Function String_ToInt( Str : String) : Integer');
15513:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15514:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15515:  Function String_Reverse( Str : String) : String');
15516:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15517:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15518:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15519:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15520:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15521:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15522:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15523:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15524:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15525:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15526:  Function String_End( Str : String; Cars : Word) : String');
15527:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15528:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15529:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15530:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15531:  Function String_IsNumbers( Str : String) : Boolean');
15532:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15533:  Function StringToCsvCell( aStr : String) : String');
15534: end;
15535:
15536: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15537: begin
15538:  Function LongDayName( aDate : TDate) : String');
15539:  Function LongMonthName( aDate : TDate) : String');
15540:  Function ShortYearOf( aDate : TDate) : byte');
15541:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15542:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15543:  Function SecondsToMinutes( Seconds : Integer) : Double');
15544:  Function MinutesToSeconds( Minutes : Double) : Integer');
15545:  Function MinutesToHours( Minutes : Integer) : Double');
15546:  Function HoursToMinutes( Hours : Double) : Integer');
15547:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15548:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15549:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15550:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15551:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15552:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15553:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15554:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15555:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15556: end;
15557:
15558: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15559: begin
15560:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15561:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15562:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15563:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15564:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15565:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15566:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15567:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15568:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15569:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15570:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15571:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15572:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15573:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15574:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15575:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15576:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15577:  Procedure cyCenterControl( aControl : TControl)');
15578:  Function GetLastParent( aControl : TControl) : TWinControl');
15579:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15580:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15581: end;
15582:
15583: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15584: begin
15585:  Function TablePackTable( Tab : TTable) : Boolean');
15586:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15587:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15588:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15589:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15590:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15591:  Function TableEmptyTable( Tab : TTable) : Boolean');
15592:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15593:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15594:  Function
       TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
```

```
15595:  Function TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15596:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15597: end;
15598:
15599: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15600: begin
15601:   SIRegister_TcyRunTimeDesign(CL);
15602:   SIRegister_TcyShadowText(CL);
15603:   SIRegister_TcyBgPicture(CL);
15604:   SIRegister_TcyGradient(CL);
15605:   SIRegister_tcyBevel(CL);
15606:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15607:   SIRegister_tcyBevels(CL);
15608:   SIRegister_TcyImagelistOptions(CL);
15609:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15610: end;
15611:
15612: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15613: begin
15614:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15615:    SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15616:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15617:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15618:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15619:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15620:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15621:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15622:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
       BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15623:  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;
15624:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15625:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15626:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
       TColor; aState : TButtonState; Focused, Hot : Boolean)');
15627:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
       GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15628:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
       const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15629:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
       TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15630:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
       TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15631:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
       WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15632:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15633:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15634:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15635:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
       CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15636:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15637:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15638:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15639:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15640:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15641:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15642:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15643:  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);
15644:  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;
15645:  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);');
15646:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15647:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15648:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15649:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15650:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15651:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15652:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15653:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15654:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15655:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15656:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
```

```
15657:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15658:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15659:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15660:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15661:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15662: end;
15663:
15664: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15665: begin
15666:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15667:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15668:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15669:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15670:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15671:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15672:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15673:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15674:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15675:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15676:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15677:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
      bmInvertReverseFromColor);
15678:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
      rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15679:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15680:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15681: end;
15682:
15683: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15684: begin
15685:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15686:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15687:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15688:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15689:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15690:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15691:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15692:  Const SC_GROUP_IDENTIFIERW','String') '+');
15693:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15694:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15695:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15696:  Const SERVICE_INACTIVE','LongWord $00000002);
15697:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15698:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15699:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15700:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15701:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15702:  Const SERVICE_STOPPED','LongWord $00000001);
15703:  Const SERVICE_START_PENDING','LongWord $00000002);
15704:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15705:  Const SERVICE_RUNNING','LongWord $00000004);
15706:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15707:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15708:  Const SERVICE_PAUSED','LongWord $00000007);
15709:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15710:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15711:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15712:  Const SC_MANAGER_CONNECT','LongWord $0001);
15713:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15714:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15715:  Const SC_MANAGER_LOCK','LongWord $0008);
15716:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15717:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15718:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15719:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15720:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15721:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15722:  Const SERVICE_START','LongWord $0010);
15723:  Const SERVICE_STOP','LongWord $0020);
15724:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15725:  Const SERVICE_INTERROGATE','LongWord $0080);
15726:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15727:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15728:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15729:  Const SERVICE_ADAPTER','LongWord $00000004);
15730:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15731:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15732:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15733:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15734:  Const SERVICE_BOOT_START','LongWord $00000000);
15735:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15736:  Const SERVICE_AUTO_START','LongWord $00000002);
15737:  Const SERVICE_DEMAND_START','LongWord $00000003);
15738:  Const SERVICE_DISABLED','LongWord $00000004);
15739:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15740:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15741:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15742:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15743:   CL.AddTypeS('SC_HANDLE', 'THandle');
```

```
15744:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15745:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15746:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15747:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15748:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15749:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15750:   Const TServiceStatus', '_SERVICE_STATUS');
15751:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15752:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15753:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15754:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15755:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15756:   TEnumServiceStatus', 'TEnumServiceStatusA');
15757:   SC_LOCK', '___Pointer');
15758:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15759:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15760:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15761:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15762:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15763:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15764:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15765:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15766:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15767:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15768:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15769:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15770:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15771:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15772:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15773:   TQueryServiceConfig', 'TQueryServiceConfigA');
15774:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15775:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15776:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15777:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15778:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15779:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15780:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15781:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
      TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15782:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
      lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
      : DWORD):BOOL');
15783:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
      lpcchBuffer:DWORD):BOOL');
15784:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
      lpcchBuffer:DWORD):BOOL;
15785:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15786:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15787:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15788:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15789:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
      cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15790:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15791:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15792:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15793:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15794: end;
15795:
15796: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15797: begin
15798:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
      AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
      BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15799:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
      DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15800:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15801:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
      TWinControl;
15802:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
      AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15803:  Function CreateNotifyThread(const
      FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15804: end;
15805:
15806: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15807: begin
15808:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15809:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15810:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15811:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15812:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15813:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15814:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15815:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15816:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15817:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15818:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
```

```
15819:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15820:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15821:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15822:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15823:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15824:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15825:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15826:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15827:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15828:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15829:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15830:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15831:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15832:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15833:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15834:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15835:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15836:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15837:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15838:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15839:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15840:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15841:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15842:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15843:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15844:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15845:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15846:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15847:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15848:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15849:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15850:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15851:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15852:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15853:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
      List:TStrings):Bool
15854:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15855:   FindClass('TOBJECT'),'EJclFileSummaryError');
15856:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15857:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15858:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15859:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15860:   SIRegister_TJclFilePropertySet(CL);
15861:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15862:   SIRegister_TJclFileSummary(CL);
15863:   SIRegister_TJclFileSummaryInformation(CL);
15864:   SIRegister_TJclDocSummaryInformation(CL);
15865:   SIRegister_TJclMediaFileSummaryInformation(CL);
15866:   SIRegister_TJclMSISummaryInformation(CL);
15867:   SIRegister_TJclShellSummaryInformation(CL);
15868:   SIRegister_TJclStorageSummaryInformation(CL);
15869:   SIRegister_TJclImageSummaryInformation(CL);
15870:   SIRegister_TJclDisplacedSummaryInformation(CL);
15871:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15872:   SIRegister_TJclMiscSummaryInformation(CL);
15873:   SIRegister_TJclWebViewSummaryInformation(CL);
15874:   SIRegister_TJclMusicSummaryInformation(CL);
15875:   SIRegister_TJclDRMSummaryInformation(CL);
15876:   SIRegister_TJclVideoSummaryInformation(CL);
15877:   SIRegister_TJclAudioSummaryInformation(CL);
15878:   SIRegister_TJclControlPanelSummaryInformation(CL);
15879:   SIRegister_TJclVolumeSummaryInformation(CL);
15880:   SIRegister_TJclShareSummaryInformation(CL);
15881:   SIRegister_TJclLinkSummaryInformation(CL);
15882:   SIRegister_TJclQuerySummaryInformation(CL);
15883:   SIRegister_TJclImageInformation(CL);
15884:   SIRegister_TJclJpegSummaryInformation(CL);
15885: end;
15886:
15887: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15888: begin
15889:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15890:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15891:   T8087Infinity','( icProjective, icAffine )');
15892:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15893:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15894:  Function Get8087ControlWord : Word');
15895:  Function Get8087Infinity : T8087Infinity');
15896:  Function Get8087Precision : T8087Precision');
15897:  Function Get8087Rounding : T8087Rounding');
15898:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15899:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15900:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15901:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15902:  Function Set8087ControlWord( const Control : Word) : Word');
15903:  Function ClearPending8087Exceptions : T8087Exceptions');
15904:  Function GetPending8087Exceptions : T8087Exceptions');
15905:  Function GetMasked8087Exceptions : T8087Exceptions');
15906:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
```

```
15907:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15908:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15909: end;
15910:
15911: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15912: begin
15913:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15914:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15915:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
       Accept:Bool;Sorted:Bool;
15916:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15917:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15918:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15919:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15920:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15921: end;
15922:
15923: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15924: begin
15925:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15926:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15927:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15928:   type ULONG', 'Cardinal');
15929:      LPCWSTR', 'PChar');
15930:   CL.AddTypeS('LPWSTR', 'PChar');
15931:   LPSTR', 'PChar');
15932:   TBindVerb', 'ULONG');
15933:   TBindInfoF', 'ULONG');
15934:   TBindF', 'ULONG');
15935:   TBSCF', 'ULONG');
15936:   TBindStatus', 'ULONG');
15937:   TCIPStatus', 'ULONG');
15938:   TBindString', 'ULONG');
15939:   TPiFlags', 'ULONG');
15940:   TOIBdgFlags', 'ULONG');
15941:   TParseAction', 'ULONG');
15942:   TPSUAction', 'ULONG');
15943:   TQueryOption', 'ULONG');
15944:   TPUAF', 'ULONG');
15945:   TSZMFlags', 'ULONG');
15946:   TUrlZone', 'ULONG');
15947:   TUrlTemplate', 'ULONG');
15948:   TZAFlags', 'ULONG');
15949:   TUrlZoneReg', 'ULONG');
15950:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15951:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15952:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15953:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15954:  const 'CF_NULL','LongInt').SetInt( 0);
15955:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15956:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15957:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15958:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15959:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15960:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15961:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15962:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15963:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15964:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15965:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15966:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15967:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15968:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15969:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15970:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15971:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15972:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15973:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15974:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15975:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15976:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15977:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15978:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15979:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15980:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15981:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15982:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15983:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15984:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15985:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15986:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15987:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15988:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15989:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15990:   SIRegister_IPersistMoniker(CL);
15991:   SIRegister_IBindProtocol(CL);
15992:   SIRegister_IBinding(CL);
15993:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15994:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
```

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

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

```
16170:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16171:   SIRegister_IInternetSession(CL);
16172:   SIRegister_IInternetThreadSwitch(CL);
16173:   SIRegister_IInternetPriority(CL);
16174:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16175:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16176:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16177:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16178:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16179:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16180:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16181:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16182:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16183:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16184:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16185:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16186:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16187:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16188:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16189:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16190:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16191:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16192:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16193:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16194:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16195:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16196:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16197:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16198:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16199:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16200:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16201:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16202:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16203:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16204:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16205:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16206:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16207:   SIRegister_IInternetProtocolInfo(CL);
16208:   IOInet', 'IInternet');
16209:   IOInetBindInfo', 'IInternetBindInfo');
16210:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16211:   IOInetProtocol', 'IInternetProtocol');
16212:   IOInetProtocolSink', 'IInternetProtocolSink');
16213:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16214:   IOInetSession', 'IInternetSession');
16215:   IOInetPriority', 'IInternetPriority');
16216:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16217:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16218:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16219:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16220:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16221:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16222:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16223:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16224:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16225:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16226:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16227:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16228:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16229:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16230:    //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16231:   //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16232: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16233: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16234:   //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16235:   //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16236:   //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16237:   const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16238:    SIRegister_IInternetSecurityMgrSite(CL);
16239:   const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16240:   const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16241:   const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16242:   const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16243:   const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16244:   const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16245:   const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16246:   const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16247:   const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16248:   const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16249:   const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
```

```
16250:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16251:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16252:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16253:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16254:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16255:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16256:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16257:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16258:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16259:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16260:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16261:  const 'PUAF_NOUIFLOCKED','LongWord').SetUInt( $00100000);
16262:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16263:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16264:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16265:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16266:   SIRegister_IInternetSecurityManager(CL);
16267:   SIRegister_IInternetHostSecurityManager(CL);
16268:   SIRegister_IInternetSecurityManagerEx(CL);
16269:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16270:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16271:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16272:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16273:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16274:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16275:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16276:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16277:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16278:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16279:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16280:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16281:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16282:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16283:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16284:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16285:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16286:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16287:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16288:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16289:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16290:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16291:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16292:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16293:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16294:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16295:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16296:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16297:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16298:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16299:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16300:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16301:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16302:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16303:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16304:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16305:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16306:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16307:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16308:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16309:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16310:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16311:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16312:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16313:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16314:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16315:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16316:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16317:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16318:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16319:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16320:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16321:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16322:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16323:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16324:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16325:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16326:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16327:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16328:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16329:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16330:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16331:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16332:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16333:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16334:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16335:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16336:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16337:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16338:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
```

```
16339:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16340:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16341:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16342:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16343:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16344:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16345:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16346:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16347:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16348:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16349:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16350:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16351:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16352:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16353:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16354:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16355:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16356:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16357:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16358:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16359:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16360:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16361:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16362:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16363:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16364:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16365:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16366:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16367:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16368:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16369:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16370:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16371:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16372:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16373:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16374:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16375:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16376:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16377:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16378:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16379:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16380:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16381:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16382:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16383:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16384:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16385:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16386:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16387:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16388:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16389:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16390:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16391:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16392:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16393:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16394:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16395:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16396:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16397:   //PZoneAttributes', '^TZoneAttributes // will not work');
16398:  _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');
16399:   { _ZONEATTRIBUTES = packed record
16400:     cbSize: ULONG;
16401:     szDisplayName: array [0..260 - 1] of WideChar;
16402:     szDescription: array [0..200 - 1] of WideChar;
16403:     szIconPath: array [0..260 - 1] of WideChar;
16404:     dwTemplateMinLevel: DWORD;
16405:     dwTemplateRecommended: DWORD;
16406:     dwTemplateCurrentLevel: DWORD;
16407:     dwFlags: DWORD;
16408:   end; }
16409:   TZoneAttributes', '_ZONEATTRIBUTES');
16410:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16411:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16412:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16413:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16414:   SIRegister_IInternetZoneManager(CL);
16415:   SIRegister_IInternetZoneManagerEx(CL);
16416:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16417:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16418:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16419:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16420:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16421:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16422:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16423:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16424:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16425:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
```

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

```
16514:  Function digitcount( n : int64) : integer');
16515:  Function nextpermute( var a : array of integer) : boolean');
16516:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16517:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16518:  Function InttoBinaryStr( nn : integer) : string');
16519:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16520:  Function AngleToStr( angle : extended) : string');
16521:  Function deg2rad( deg : extended) : extended');
16522:  Function rad2deg( rad : extended) : extended');
16523:  Function GetLongToMercProjection( const long : extended) : extended');
16524:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16525:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16526:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16527:   SIRegister_TPrimes(CL);
16528:   //RIRegister_TPrimes(CL);
16529:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16530:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16531:  Function Random64( const N : Int64) : Int64;');
16532:  Procedure Randomize64');
16533:  Function Random641 : extended;');
16534:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16535: end;
16536:
16537: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16538: begin
16539:   TrealPoint', 'record x : extended; y : extended; end');
16540:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16541:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16542:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16543:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16544:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16545:  Function realpoint( x, y : extended) : TRealPoint');
16546:  Function dist( const p1, p2 : TrealPoint) : extended');
16547:  Function intdist( const p1, p2 : TPoint) : integer');
16548:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16549:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16550:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16551:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16552:  Function GetTheta( const L : TLine) : extended;');
16553:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16554:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16555:  Procedure Extendline( var L : TLine; dist : integer);');
16556:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16557:  Function Linesintersect( line1, line2 : TLine) : boolean');
16558:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16559:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16560:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16561:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16562:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16563:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16564:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16565:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16566:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16567:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16568:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16569:  Function DegtoRad( d : extended) : extended');
16570:  Function RadtoDeg( r : extended) : extended');
16571:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16572:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16573:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16574:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16575:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16576:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16577:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16578:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16579:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16580: end;
16581:
16582:
16583: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16584: begin
16585:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16586:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16587:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16588:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16589:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16590:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16591:    +'arth : extended; Phase : extended; end');
16592:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16593:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16594:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16595:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16596:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16597:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
```

```
16598:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16599:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16600:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16601:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16602:   SIRegister_TAstronomy(CL);
16603:  Function AngleToStr( angle : extended) : string');
16604:  Function StrToAngle( s : string; var angle : extended) : boolean');
16605:  Function HoursToStr24( t : extended) : string');
16606:  Function RPoint( x, y : extended) : TRPoint');
16607:  Function getStimename( t : TDTType) : string');
16608: end;
16609:
16610: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16611: begin
16612:   TCardValue', 'Integer');
16613:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16614:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16615:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16616:   SIRegister_TCard(CL);
16617:   SIRegister_TDeck(CL);
16618: end;
16619:
16620: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16621: begin
16622:   tMethodCall', 'Procedure');
16623:   tVerboseCall', 'Procedure ( s : string)');
16624:  // PTEdge', '^TEdge // will not work');
16625:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16626:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16627:   SIRegister_TNode(CL);
16628:   SIRegister_TGraphList(CL);
16629: end;
16630:
16631: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16632: begin
16633:   ParserFloat', 'extended');
16634:   //PParserFloat', '^ParserFloat // will not work');
16635:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16636:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16637:   //POperation', '^TOperation // will not work');
16638:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16639:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16640:    +'; Token : TDFFToken; end');
16641:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16642:   (CL.FindClass('TOBJECT'),'EMathParserError');
16643:   CL.FindClass('TOBJECT'),'ESyntaxError');
16644:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16645:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16646:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16647:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16648:   (CL.FindClass('TOBJECT'),'EBadName');
16649:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16650:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16651:   SIRegister_TCustomParser(CL);
16652:   SIRegister_TExParser(CL);
16653: end;
16654:
16655:   function isService: boolean;
16656:   begin
16657:     result:= NOT(Application is TApplication);
16658:     {result:= Application is TServiceApplication;}
16659:   end;
16660:   function isApplication: boolean;
16661:   begin
16662:     result:= Application is TApplication;
16663:   end;
16664:   //SM_REMOTESESSION = $1000
16665:   function isTerminalSession: boolean;
16666:   begin
16667:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16668:   end;
16669:
16670: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16671: begin
16672:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16673:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16674:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16675:  Function cyURLEncode( const S : string) : string');
16676:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16677:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16678:  Function cyColorToHtml( aColor : TColor) : String');
16679:  Function HtmlToColor( aHtmlColor : String) : TColor');
16680:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16681:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16682:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16683:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16684:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
```

```
16685:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16686:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16687:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16688:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16689:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16690:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16691:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16692: end;
16693:
16694:
16695: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16696: begin
16697:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16698:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16699:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16700:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16701:    +'inationsRepeat, CombinationsRepeatDown )');
16702:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16703:   SIRegister_TComboSet(CL);
16704: end;
16705:
16706:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16707: begin
16708:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16709:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16710:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16711:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16712:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16713:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16714:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16715:   SIRegister_TcyBaseComm(CL);
16716:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16717:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16718:  Function ValidateFileMappingName( aName : String) : String');
16719:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16720: end;
16721:
16722: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16723: begin
16724:   CL.AddTypeS('DERString', 'String');
16725:   CL.AddTypeS('DERChar', 'Char');
16726:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16727:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16728:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16729:   CL.AddTypeS('DERNString', 'String');
16730:  const DERDecimalSeparator','String').SetString( '.');
16731:  const DERDefaultChars','String')('+°@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16732:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16733:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16734:  Function isValidWebMailChar( aChar : Char) : Boolean');
16735:  Function isValidwebSite( aStr : String) : Boolean');
16736:  Function isValidWebMail( aStr : String) : Boolean');
16737:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16738:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16739:  Function IsDERChar( aChar : Char) : Boolean');
16740:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16741:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16742:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16743:  Function IsDERSymbols( aDERString : String) : Boolean');
16744:  Function StringToDERCharSet( aStr : String) : DERString');
16745:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16746:  Function IsDERNChar( aChar : Char) : Boolean');
16747:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16748:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16749:  Function DERExtractWebMail( aDERStr : DERString) : String');
16750:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16751:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16752:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
      RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16753:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
      aElementsType : TElementsType) : String;');
16754:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
      Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16755: end;
16756:
16757: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16758: begin
16759:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16760:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16761:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16762:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16763:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16764:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16765:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16766:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16767:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
      RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
```

```
16768:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
        PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
        PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16769:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
        SingleDestinationColor, RefreshBmp : Boolean)');
16770:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16771:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16772:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16773:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16774:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16775:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16776:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16777:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16778: end;
16779:
16780: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16781: begin
16782:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16783:   TPCMChannel', '( cMono, cStereo )');
16784:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16785:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16786:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16787:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16788:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16789:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16790:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16791:    +'it48000Hz, Stereo16bit48000Hz )');
16792:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16793:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16794:    tWaveFormatEx', 'PWaveFormatEx');
16795:    HMMIO', 'Integer');
16796:   TWaveDeviceFormats', 'set of TPCMFormat');
16797:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16798:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16799:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16800:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16801:   TWaveOutOptions', 'set of TWaveOutOption');
16802:   TStreamOwnership2', '( soReference, soOwned )');
16803:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16804:  // PRawWave', '^TRawWave // will not work');
16805:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16806:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16807:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
16808:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16809:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16810:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16811:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16812:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16813:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16814:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16815:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16816:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16817:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16818:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16819:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16820:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16821:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16822:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16823:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16824:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16825:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16826:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16827:  GetAudioFormat( FormatTag : Word) : String');
16828:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16829:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16830:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16831:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16832:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16833:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16834:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16835:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
        TObject; BufferSize : DWORD) : Boolean');
16836:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
        dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16837:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
        TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16838:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16839:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16840:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16841:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16842:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16843:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16844: end;
16845:
16846: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16847: begin
16848:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16849:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16850:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
```

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

```
16938:  Procedure SetVistaContentFonts( const AFont : TFont)');
16939:  Function GetProductType( var sType : String) : Boolean');
16940:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16941:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16942:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16943:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16944:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16945:  Function lstrlen( lpString : PChar) : Integer');
16946:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16947:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16948: end;
16949:
16950: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16951: begin
16952:  'cLineTerminator','Char').SetString( #10);
16953:  'cLineTerminators','String').SetString( #13#10);
16954:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16955:   SIRegister_TFixedCriticalSection(CL);
16956:   SIRegister_TMultiReadSingleWrite(CL);
16957:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16958:  Function GetDecimalSeparator : Char');
16959:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16960:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16961:   CL.AddTypeS('NativeInt', 'Integer');
16962:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16963:   CL.AddTypeS('NativeUInt', 'Cardinal');
16964:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16965:   //CL.AddTypeS('TBytes', 'array of Byte');
16966:   CL.AddTypeS('RawByteString', 'UnicodeString');
16967:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16968:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16969:   SIRegister_TPath(CL);
16970:   SIRegister_TFile(CL);
16971:   SIRegister_TdwsThread(CL);
16972:  Function GetSystemMilliseconds : Int64');
16973:  Function UTCDateTime : TDateTime');
16974:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16975:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16976:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16977:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16978:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16979:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16980:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16981:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16982:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16983:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16984:  Function InterlockedIncrement( var val : Integer) : Integer');
16985:  Function InterlockedDecrement( var val : Integer) : Integer');
16986:  Procedure FastInterlockedIncrement( var val : Integer)');
16987:  Procedure FastInterlockedDecrement( var val : Integer)');
16988:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16989:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16990:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16991:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16992:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16993:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16994:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16995:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16996:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16997:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16998:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16999:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
17000:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
17001:  Procedure CloseFileHandle( hFile : THandle)');
17002:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
17003:  Function FileMove( const existing, new : UnicodeString) : Boolean');
17004:  Function dwsFileDelete( const fileName : String) : Boolean');
17005:  Function FileRename( const oldName, newName : String) : Boolean');
17006:  Function dwsFileSize( const name : String) : Int64');
17007:  Function dwsFileDateTime( const name : String) : TDateTime');
17008:  Function DirectSet8087CW( newValue : Word) : Word');
17009:  Function DirectSetMXCSR( newValue : Word) : Word');
17010:  Function TtoObject( const T: byte) : TObject');
17011:  Function TtoPointer( const T: byte) : ___Pointer');
17012:  Procedure GetMemForT(var T: byte; Size : integer)');
17013:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
17014: end;
17015:
17016: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
17017: begin
17018:  'IPStrSize','LongInt').SetInt( 15);
17019:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17020:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
17021:  'ADWSBASE','LongInt').SetInt( 9000);
17022:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17023:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
```

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

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

```
17196:   CL.AddTypeS('TPreviousFsRedirectionState', record DidDisable : Boolean; OldValue : ___Pointer; end');
17197:  CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17198:  Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Boo;
17199:  Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17200:  Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17201:  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;
17202:  Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
        FailIfExists : BOOL) : BOOL');
17203:  Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17204:  Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17205:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17206:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
        TWin32FindData) : THandle');
17207:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17208:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17209:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
        TFileVersionNumbers) : Boolean');
17210:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17211:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17212:  Function MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilen,NewFilename:String;const
        Flags:DWORD):BOOL;
17213:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17214:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17215:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17216:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17217:   SIRegister_TFileRedir(CL);
17218:   SIRegister_TTextFileReaderRedir(CL);
17219:   SIRegister_TTextFileWriterRedir(CL);
17220: end;
17221:
17222: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17223: begin
17224:   //CL.AddTypeS('LongWord', 'Cardinal');
17225:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17226:  Function Compare64( const N1, N2 : Integer64) : Integer');
17227:  Procedure Dec64( var X : Integer64; N : LongWord)');
17228:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17229:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17230:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
17231:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17232:  Function Integer64ToStr( X : Integer64) : String');
17233:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17234:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17235:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17236:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17237:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17238: end;
17239:
17240: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17241: begin
17242:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17243:   SIRegister_TSimpleStringList(CL);
17244:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17245:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17246:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17247:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17248:  // TMD5Digest = array[0..15] of Byte;
17249:   //     TSHA1Digest = array[0..19] of Byte;
17250:  Function CheckForMutexes( Mutexes : String) : Boolean');
17251:  Function CreateTempDir : String');
17252:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17253:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
        FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17254:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
        DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
        TDeleteFileProc; const Param : Pointer) : Boolean');
17255:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
        TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
        TDetermineDefaultLanguageResult');
17256:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17257:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17258:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17259:  Function GetComputerNameString : String');
17260:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17261:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17262:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17263: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17264:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17265:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17266: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17267:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17268:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
        TotBytes:Int64):Bool;
17269:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
        TotalBytes: Integer64):Bool;
```

```
17270:  Function GetUserNameString : String');
17271:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17272:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
        String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
        ResultCode:Integer) : Boolean');
17273: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
        TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17274:  Procedure InternalError( const Id : String)');
17275:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17276:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17277:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17278:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17279:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17280:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
17281:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17282:  Procedure RefreshEnvironment');
17283:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17284:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17285:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17286:  Function RestartComputer : Boolean');
17287:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17288:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17289:  Procedure Win32ErrorMsg( const FunctionName : String)');
17290:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17291:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17292:  //from Func2
17293:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
        IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
17294:   //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17295:  Procedure RegisterTypeLibrary( const Filename : String)');
17296:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17297:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17298:  function getVersionInfoEx3: TOSVersionInfoEx;');
17299:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17300:  procedure InitOle;');
17301:  Function ExpandConst( const S : String) : String');
17302:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17303:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17304:  Function ExpandConstIfPrefixed( const S : String) : String');
17305:  Procedure LogWindowsVersion');
17306:  Function EvalCheck( const Expression : String) : Boolean');
17307: end;
17308:
17309: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17310: begin
17311:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17312:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17313:   SIRegister_TResourceModule(CL);
17314:   SIRegister_TResourceDetails(CL);
17315:   SIRegister_TAnsiResourceDetails(CL);
17316:   SIRegister_TUnicodeResourceDetails(CL);
17317:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17318:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17319:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17320:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17321:  Function ResourceNameToInt( const s : string) : Integer');
17322:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17323: end;
17324:
17325:
17326: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17327: begin
17328:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17329:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
17330:     RegisterMethod('Constructor Create');
17331:     RegisterMethod('Procedure Free');
17332:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17333:     RegisterMethod('Procedure WriteString( const S : String)');
17334:     RegisterMethod('Procedure ReadString( var S : String)');
17335:   end;
17336:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17337:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17338:       SimpleComPort.WriteString(AsciiChar);
17339: end;
17340:
17341:
17342: procedure SIRegister_Console(CL: TPSPascalCompiler);
17343: begin
17344:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17345: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17346:  ('conBW40','LongInt').SetInt( 0);
17347:  ('conCO40','LongInt').SetInt( 1);
17348:  ('conBW80','LongInt').SetInt( 2);
17349:  ('conCO80','LongInt').SetInt( 3);
17350:  ('conMono','LongInt').SetInt( 7);
17351:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17352:  //CL.AddConstantN('C40','').SetString( CO40);
17353:  //CL.AddConstantN('C80','').SetString( CO80);
17354:  Function conReadKey : Char');
```

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

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

```
17532:   SIRegister_TcyCopyFiles(CL);
17533:  Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
       ResetAttr:boolean): TCopyFileResult');
17534:  Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
       TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17535:  Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
        ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;'
17536:                       + FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17537: end;
17538:
17539: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17540: begin
17541:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17542:   SIRegister_TcyFileAttributes(CL);
17543:   SIRegister_TSearchRecInstance(CL);
17544:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17545:   TOptions', 'set of TOption');
17546:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17547:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
       Accept:bool;
17548:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17549:   SIRegister_TcyCustomSearchFiles(CL);
17550:   SIRegister_TcySearchFiles(CL);
17551:  Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17552:  Function IscyFolder( aSRec : TSearchrec) : Boolean');
17553: end;
17554:
17555: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17556: begin
17557:  Function jCountChar( const s : string; ch : char) : integer');
17558:  Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings)');
17559:  Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17560:  Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17561:  Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17562:  Function jNormalizeDateSeparator( const s : string) : string');
17563:  Function jIsValidDateString( const Value : string) : boolean');
17564:  Function jIsValidTimeString( const Value : string) : boolean');
17565:  Function jIsValidDateTimeString( const Value : string) : boolean');
17566: end;
17567:
17568: procedure SIRegister_kcMapViewer(CL: TPSPascalCompiler);
17569: begin
17570:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapViewer');
17571:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
17572:    +'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
17573:    +', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
17574:    +'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
17575:    +'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17576:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17577:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17578:   TIntPoint', 'record X : Int64; Y : Int64; end');
17579:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17580:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17581:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17582:   SIRegister_TCustomDownloadEngine(CL);
17583:   SIRegister_TCustomGeolocationEngine(CL);
17584:   SIRegister_TMapViewer(CL);
17585: end;
17586:
17587: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17588: begin
17589:  (*CL.AddDelphiFunction('Function isFunc( name : TNamePart) : Boolean');*)
17590:  CL.AddDelphiFunction('Function isUnnamedFunc( name : TNamepart) : Boolean');
17591:  Function isPtrToFunc( name : TNamePart) : Boolean');
17592:  Function isFuncRetFuncPtr( name : TNamePart) : Boolean');
17593:  Function isPtrToFuncRetFuncPtr( name : TNamePart) : Boolean');
17594:  Function GetFuncParam( name : TNamePart) : TNamePart');
17595:  Function isArray( name : TNamePart) : Boolean');
17596:  Function GetArrayPart( name : TNamePart) : TNamePart');
17597:  Function GetIdFromPart( name : TNamePart) : AnsiString');
17598:  Function GetIdPart( name : TNamePart) : TNamePart');
17599:  Function isNamePartPtrToFunc( part : TNamePart) : Boolean');
17600:  Function isAnyBlock( part : TNamePart) : Boolean');*)
17601:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17602:   SIRegister_TLineBreaker(CL);
17603:   CL.AddTypeS('TNameKind', 'Integer');
17604:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17605:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17606:  Function SphericalMod( X : Extended) : Extended');
17607:  Function cSign( Value : Extended) : Extended');
17608:  Function LimitFloat( const eValue, eMin, eMax : Extended) : Extended');
17609:  Function AngleToRadians( iAngle : Extended) : Extended');
17610:  Function RadiansToAngle( eRad : Extended) : Extended');
17611:  Function Cross180( iLong : Double) : Boolean');
17612:  Function Mod180( Value : integer) : Integer');
17613:  Function Mod180Float( Value : Extended) : Extended');
17614:  Function MulDivFloat( a, b, d : Extended) : Extended');
17615:  Function LongDiff( iLong1, iLong2 : Double) : Double');
17616:  Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
```

```
17617:  Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17618:  Function FixFilePath( const Inpath, CheckPath : string) : string');
17619:  Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17620:  Procedure FillStringList( sl : TStringList; const aText : string)');
17621: end;
17622:
17623: procedure SIRegister_LedNumber(CL: TPSPascalCompiler);
17624: begin
17625:   TLedSegmentSize', 'Integer');
17626:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
17627:   SIRegister_TCustomLEDNumber(CL);
17628:   SIRegister_TLEDNumber(CL);
17629: end;
17630:
17631: procedure SIRegister_StStrL(CL: TPSPascalCompiler);
17632: begin
17633:   CL.AddTypeS('LStrRec', 'record AllocSize : Longint; RefCount : Longint; Length : Longint; end');
17634:    CL.AddTypeS('AnsiChar', 'Char');
17635:     CL.AddTypeS('BTable', 'array[0..255] of Byte');       //!!!
17636:   CL.AddDelphiFunction('Function HexBL( B : Byte) : AnsiString');
17637:  Function HexWL( W : Word) : AnsiString');
17638:  Function HexLL( L : LongInt) : AnsiString');
17639:  Function HexPtrL( P : ___Pointer) : AnsiString');
17640:  Function BinaryBL( B : Byte) : AnsiString');
17641:  Function BinaryWL( W : Word) : AnsiString');
17642:  Function BinaryLL( L : LongInt) : AnsiString');
17643:  Function OctalBL( B : Byte) : AnsiString');
17644:  Function OctalWL( W : Word) : AnsiString');
17645:  Function OctalLL( L : LongInt) : AnsiString');
17646:  Function Str2Int16L( const S : AnsiString; var I : SmallInt) : Boolean');
17647:  Function Str2WordL( const S : AnsiString; var I : Word) : Boolean');
17648:  Function Str2LongL( const S : AnsiString; var I : LongInt) : Boolean');
17649:  Function Str2RealL( const S : AnsiString; var R : Double) : Boolean');
17650:  Function Str2RealL( const S : AnsiString; var R : Real) : Boolean');
17651:  Function Str2ExtL( const S : AnsiString; var R : Extended) : Boolean');
17652:  Function Long2StrL( L : LongInt) : AnsiString');
17653:  Function Real2StrL( R : Double; Width : Byte; Places : ShortInt) : AnsiString');
17654:  Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : AnsiString');
17655:  Function ValPrepL( const S : AnsiString) : AnsiString');
17656:  Function CharStrL( C : Char; Len : Cardinal) : AnsiString');
17657:  Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17658:  Function PadLL( const S : AnsiString; Len : Cardinal) : AnsiString');
17659:  Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17660:  Function LeftPadL( const S : AnsiString; Len : Cardinal) : AnsiString');
17661:  Function TrimLeadL( const S : AnsiString) : AnsiString');
17662:  Function TrimTrailL( const S : AnsiString) : AnsiString');
17663:  Function TrimL( const S : AnsiString) : AnsiString');
17664:  Function TrimSpacesL( const S : AnsiString) : AnsiString');
17665:  Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17666:  Function CenterL( const S : AnsiString; Len : Cardinal) : AnsiString');
17667:  Function EntabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17668:  Function DetabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17669:  Function ScrambleL( const S, Key : AnsiString) : AnsiString');
17670:  Function SubstituteL( const S, FromStr, ToStr : AnsiString) : AnsiString');
17671:  Function FilterL( const S, Filters : AnsiString) : AnsiString');
17672:  Function CharExistsL( const S : AnsiString; C : AnsiChar) : Boolean');
17673:  Function CharCountL( const S : AnsiString; C : AnsiChar) : Cardinal');
17674:  Function WordCountL( const S, WordDelims : AnsiString) : Cardinal');
17675:  Function WordPositionL( N : Cardinal; const S, WordDelims : AnsiString; var Pos : Cardinal) : Boolean');
17676:  Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString) : AnsiString');
17677:  Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar) : Cardinal');
17678:  Function AsciiPositionL(N: Cardinal;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17679:  Function ExtractAsciiL( N : Cardinal; const S, WordDelims : AnsiString; Quote : AnsiChar) : AnsiString');
17680:  Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17681:  Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17682:  Function CompStringL( const S1, S2 : AnsiString) : Integer');
17683:  Function CompUCStringL( const S1, S2 : AnsiString) : Integer');
17684:  Function SoundexL( const S : AnsiString) : AnsiString');
17685:  Function MakeLetterSetL( const S : AnsiString) : Longint');
17686:  Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable)');
17687:  Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
       Pos:Card):Bool;
17688:  Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
       var Pos : Cardinal) : Boolean');
17689:  Function DefaultExtensionL( const Name, Ext : AnsiString) : AnsiString');
17690:  Function ForceExtensionL( const Name, Ext : AnsiString) : AnsiString');
17691:  Function JustFilenameL( const PathName : AnsiString) : AnsiString');
17692:  Function JustNameL( const PathName : AnsiString) : AnsiString');
17693:  Function JustExtensionL( const Name : AnsiString) : AnsiString');
17694:  Function JustPathnameL( const PathName : AnsiString) : AnsiString');
17695:  Function AddBackSlashL( const DirName : AnsiString) : AnsiString');
17696:  Function CleanPathNameL( const PathName : AnsiString) : AnsiString');
17697:  Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal) : Boolean');
17698:  Function CommaizeL( L : LongInt) : AnsiString');
17699:  Function CommaizeChL( L : Longint; Ch : AnsiChar) : AnsiString');
17700:  Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr,RtCurr:AnsiString;Sep,
       DecPt:Char):AnsiString;
17701:  Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
       RtCurr:AnsiString;Sep:Char):AnsiString);
```

```
17702:  Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal) : Boolean');
17703:  Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal) : Boolean');
17704:  Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17705:  Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal) : AnsiString');
17706:  Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal) : AnsiString');
17707:  Function StrChDeleteL( const S : AnsiString; Pos : Cardinal) : AnsiString');
17708:  Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17709:  Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17710:  Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17711:  Function CopyLeftL( const S : AnsiString; Len : Cardinal) : AnsiString');
17712:  Function CopyMidL( const S : AnsiString; First, Len : Cardinal) : AnsiString');
17713:  Function CopyRightL( const S : AnsiString; First : Cardinal) : AnsiString');
17714:  Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal) : AnsiString');
17715:  Function CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
       SubString:AString):Bool;
17716:  Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
       SubString:AnsiString):Bool;
17717:  Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean) : AnsiString');
17718:  Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
       var SubString : AnsiString) : Boolean');
17719:  Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
       SubString : AnsiString) : Boolean');
17720:  Function DeleteWithinL( const S, Delimiter : AnsiString) : AnsiString');
17721:  Function ExtractTokensL(const S,
       Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17722:  Function IsChAlphaL( C : AnsiChar) : Boolean');
17723:  Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17724:  Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17725:  Function IsStrAlphaL( const S : AnsiString) : Boolean');
17726:  Function IsStrNumericL( const S, Numbers : AnsiString) : Boolean');
17727:  Function IsStrAlphaNumericL( const S, Numbers : AnsiString) : Boolean');
17728:  Function KeepCharsL( const S, Chars : AnsiString) : AnsiString');
17729:  Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal) : Boolean');
17730:  Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal) : Boolean');
17731:  Function LastStringL( const S, AString : AnsiString; var Position : Cardinal) : Boolean');
17732:  Function LeftTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17733:  Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Card;var
       Replacements:Card):AnsiString;
17734:  Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
       Replacements:Cardinal):AnsiString');
17735:  Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
       Replacements:Cardinal):AnsiString;
17736:  Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17737:  Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17738:  Function RightTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17739:  Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal) : boolean');
17740:  Function TrimCharsL( const S, Chars : AnsiString) : AnsiString');
17741:  Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17742:  Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17743:  end;
17744:
17745:  procedure SIRegister_pwnative_out(CL: TPSPascalCompiler);
17746: begin
17747:  CL.AddConstantN('STDIN','LongInt').SetInt( 0);
17748:  ('STDOUT','LongInt').SetInt( 1);
17749:  ('STDERR','LongInt').SetInt( 2);
17750:  Procedure NativeWrite( s : astr);');
17751:  Procedure NativeWrite1( PString : PChar);');
17752:  Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal);');
17753:  Procedure NativeWriteLn( s : astr);');
17754:  Procedure NativeWriteLn1;');
17755: end;
17756:
17757: procedure SIRegister_synwrap1(CL: TPSPascalCompiler);
17758: begin
17759:   CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse
17760:    : integer; UltimateURL : ansistring; Headers : ansistring; end');
17761:   CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17762:  Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean): TSynwInfo;');
17763:  Function GetHttpFile1( const Url, UserAgent, Outfile : string) : TSynwInfo;');
17764:  Function GetHttpFile2( const Url, Outfile : string) : TSynwInfo;');
17765:  Function GetHttpFile3( const Url, outfile : string; verbose : boolean) : TSynwInfo;');
17766:  Function GetHtm( const Url : string) : string;');
17767:  Function GetHtm1( const Url, UserAgent : string) : string;');
17768:  Function GetUrl( const Url : string; verbose : boolean) : TSynwInfo;');
17769:  Function GetUrl1( const Url, useragent : string) : TSynwInfo;');
17770:  Function GetUrl2( const Url : string) : TSynwInfo;');
17771:  Function GetUrl3( const Url : string; const http : THTTPSend; verbose : boolean): TUrlInfo;');
17772:  Function GetUrl4( const Url : string; const http : THTTPSend) : TUrlInfo;');
17773:  Procedure StrToStream( s : String; strm : TMemoryStream)');
17774:  Function StrLoadStream( strm : TStream) : String');
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: procedure SIRegister_BigIni(CL: TPSPascalCompiler);
18265: begin
18266:  CL.AddConstantN('IniTextBufferSize','LongWord').SetUInt( $7000);
18267:  CL.AddConstantN('cIniCount','String').SetString( 'Count');
18268:  TEraseSectionCallback', 'Function ( const sectionName : string; '
18269:   const sl1, sl2 : TStringList) : Boolean');
18270:   SIRegister_TCommaSeparatedInfo(CL);
18271:   SIRegister_TSectionList(CL);
18272:   SIRegister_TBigIniFile(CL);
18273:   SIRegister_TBiggerIniFile(CL);
18274:   SIRegister_TAppIniFile(CL);
18275:   SIRegister_TLibIniFile(CL);
18276:  Function ModuleName( getLibraryName : Boolean) : String');
18277: end;
18278:
18279: procedure SIRegister_ShellCtrls(CL: TPSPascalCompiler);
18280: begin
18281:   CL.AddTypeS('TRoot', 'string');
18282:   CL.AddTypeS('TRootFolder', '( rfDesktop, rfMyComputer, rfNetwork, rfRecycleBi'
18283:    +'n, rfAppData, rfCommonDesktopDirectory, rfCommonPrograms, rfCommonStartMen'
18284:    +'u, rfCommonStartup, rfControlPanel, rfDesktopDirectory, rfFavorites, rfFon'
18285:    +'ts, rfInternet, rfPersonal, rfPrinters, rfPrintHood, rfPrograms, rfRecent,'
18286:    +' rfSendTo, rfStartMenu, rfStartup, rfTemplates )');
18287:   TShellFolderCapability', '( fcCanCopy, fcCanDelete, fcCanLink, f'
18288:    +'cCanMove, fcCanRename, fcDropTarget, fcHasPropSheet )');
18289:   TShellFolderCapabilities', 'set of TShellFolderCapability');
18290:   TShellFolderProperty', '( fpCut, fpIsLink, fpReadOnly, fpShared,'
18291:    +' fpFileSystem, fpFileSystemAncestor, fpRemovable, fpValidate )');
18292:   TShellFolderProperties', 'set of TShellFolderProperty');
18293:   TShellObjectType', '( otFolders, otNonFolders, otHidden )');
18294:   TShellObjectTypes', 'set of TShellObjectType');
18295:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidPath');
18296:   SIRegister_IShellCommandVerb(CL);
18297:   SIRegister_TShellFolder(CL);
18298:   TNotifyFilter', '( nfFileNameChange, nfDirNameChange, nfAttribut'
18299:    +'eChange, nfSizeChange, nfWriteChange, nfSecurityChange )');
18300:   TNotifyFilters', 'set of TNotifyFilter');
18301:   SIRegister_TShellChangeThread(CL);
18302:   SIRegister_TCustomShellChangeNotifier(CL);
18303:   SIRegister_TShellChangeNotifier(CL);
18304:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomShellComboBox');
18305:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomShellListView');
18306:   TAddFolderEvent', 'Procedure ( Sender : TObject; AFolder : TShel'
18307:    +'lFolder; var CanAdd : Boolean)');
18308:   TGetImageIndexEvent', 'Procedure ( Sender : TObject; Index : Int'
18309:    +'eger; var ImageIndex : Integer)');
```

```
18310:   SIRegister_TCustomShellTreeView(CL);
18311:   SIRegister_TShellTreeView(CL);
18312:   SIRegister_TCustomShellComboBox(CL);
18313:   SIRegister_TShellComboBox(CL);
18314:   SIRegister_TCustomShellListView(CL);
18315:   SIRegister_TShellListView(CL);
18316:  Procedure InvokeContextMenu( Owner : TWinControl; AFolder : TShellFolder; X, Y : Integer)');
18317: end;
18318:
18319: procedure SIRegister_fmath(CL: TPSPascalCompiler);
18320: begin
18321:   CL.AddTypeS('Float', 'Double');
18322:  'FN_OK','LongInt').SetInt( 0);
18323:  CL.AddConstantN('FN_DOMAIN','LongInt').SetInt( - 1);
18324:  'FN_SING','LongInt').SetInt( - 2);
18325:  'FN_OVERFLOW','LongInt').SetInt( - 3);
18326:  'FN_UNDERFLOW','LongInt').SetInt( - 4);
18327:  'FN_TLOSS','LongInt').SetInt( - 5);
18328:  'FN_PLOSS','LongInt').SetInt( - 6);
18329:  //CL.AddConstantN('NFACT','LongInt').SetInt( 33);
18330:  CL.AddDelphiFunction('Function MathError : Integer');
18331:  Function FMin( X, Y : Float) : Float');
18332:  Function FMax( X, Y : Float) : Float');
18333:  Function IMin( X, Y : Integer) : Integer');
18334:  Function IMax( X, Y : Integer) : Integer');
18335:  Function FSgn( X : Float) : Integer');
18336:  Function Sgn0( X : Float) : Integer');
18337:  Function DSgn( A, B : Float) : Float');
18338:  Procedure FSwap( var X, Y : Float)');
18339:  Procedure ISwap( var X, Y : Integer)');
18340:  Function fExpo( X : Float) : Float');
18341:  Function fExp2( X : Float) : Float');
18342:  Function fExp10( X : Float) : Float');
18343:  Function fLog( X : Float) : Float');
18344:  Function fLog2( X : Float) : Float');
18345:  Function fLog10( X : Float) : Float');
18346:  Function fLogA( X, A : Float) : Float');
18347:  Function fIntPower( X : Float; N : Integer) : Float');
18348:  Function fPower( X, Y : Float) : Float');
18349:  Function Pythag( X, Y : Float) : Float');
18350:  Function FixAngle( Theta : Float) : Float');
18351:  Function fTan( X : Float) : Float');
18352:  Function fArcSin( X : Float) : Float');
18353:  Function fArcCos( X : Float) : Float');
18354:  Function fArcTan2( Y, X : Float) : Float');
18355:  Procedure fSinCos( X : Float; var SinX, CosX : Float)');
18356:  Function fSinh( X : Float) : Float');
18357:  Function fCosh( X : Float) : Float');
18358:  Function fTanh( X : Float) : Float');
18359:  Function fArcSinh( X : Float) : Float');
18360:  Function fArcCosh( X : Float) : Float');
18361:  Function fArcTanh( X : Float) : Float');
18362:  Procedure fSinhCosh( X : Float; var SinhX, CoshX : Float)');
18363:  Function fFact( N : Integer) : Float');
18364:  Function fBinomial( N, K : Integer) : Float');
18365:  Function fGamma( X : Float) : Float');
18366:  Function fSgnGamma( X : Float) : Integer');
18367:  Function LnGamma( X : Float) : Float');
18368:  Function fIGamma( A, X : Float) : Float');
18369:  Function fJGamma( A, X : Float) : Float');
18370:  Function fBeta( X, Y : Float) : Float');
18371:  Function fIBeta( A, B, X : Float) : Float');
18372:  Function fErf( X : Float) : Float');
18373:  Function fErfc( X : Float) : Float');
18374:  Function fPBinom(N: Integer; P: Float; K : Integer) : Float');
18375:  Function FBinom(N: Integer; P: Float; K : Integer) : Float');
18376:  Function PPoisson( Mu : Float; K : Integer) : Float');
18377:  Function FPoisson( Mu : Float; K : Integer) : Float');
18378:  Function fDNorm( X : Float) : Float');
18379:  Function FNorm( X : Float) : Float');
18380:  Function PNorm( X : Float) : Float');
18381:  Function InvNorm( P : Float) : Float');
18382:  Function fDStudent( Nu : Integer; X : Float) : Float');
18383:  Function FStudent( Nu : Integer; X : Float) : Float');
18384:  Function PStudent( Nu : Integer; X : Float) : Float');
18385:  Function fDKhi2( Nu : Integer; X : Float) : Float');
18386:  Function FKhi2( Nu : Integer; X : Float) : Float');
18387:  Function PKhi2( Nu : Integer; X : Float) : Float');
18388:  Function fDSnedecor( Nu1, Nu2 : Integer; X : Float) : Float');
18389:  Function FSnedecor( Nu1, Nu2 : Integer; X : Float) : Float');
18390:  Function PSnedecor( Nu1, Nu2 : Integer; X : Float) : Float');
18391:  Function fDExpo( A, X : Float) : Float');
18392:  Function FExpo( A, X : Float) : Float');
18393:  Function fDBeta( A, B, X : Float) : Float');
18394:  Function FBeta( A, B, X : Float) : Float');
18395:  Function fDGamma( A, B, X : Float) : Float');
18396:  Function FGamma( A, B, X : Float) : Float');
18397:  Procedure RMarIn( Seed1, Seed2 : Integer)');
18398:  Function IRanMar : LongInt');
```

```
18399:  Function RanMar : Float');
18400:  Function RanGaussStd : Float');
18401:  Function RanGauss( Mu, Sigma : Float) : Float');
18402: end;
18403:
18404: procedure SIRegister_fcomp(CL: TPSPascalCompiler);
18405: begin
18406:   CL.AddTypeS('ComplexForm', '( Rec, Pol )');
18407:   CL.AddTypeS('TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R :'
18408:    +' Float; Theta : Float; end');
18409:  Procedure CSet( var Z : TComplex; A, B : Float; F : ComplexForm)');
18410:  Procedure CConvert( var Z : TComplex; F : ComplexForm)');
18411:  Procedure CSwap( var X, Y : TComplex)');
18412:  Function CReal( Z : TComplex) : Float');
18413:  Function CImag( Z : TComplex) : Float');
18414:  Function CAbs( Z : TComplex) : Float');
18415:  Function CArg( Z : TComplex) : Float');
18416:  Function CSgn( Z : TComplex) : Integer');
18417:  Procedure CNeg( A : TComplex; var Z : TComplex)');
18418:  Procedure CConj( A : TComplex; var Z : TComplex)');
18419:  Procedure CAdd( A, B : TComplex; var Z : TComplex)');
18420:  Procedure CSub( A, B : TComplex; var Z : TComplex)');
18421:  Procedure CDiv( A, B : TComplex; var Z : TComplex)');
18422:  Procedure CMult( A, B : TComplex; var Z : TComplex)');
18423:  Procedure CLn( A : TComplex; var Z : TComplex)');
18424:  Procedure CExp( A : TComplex; var Z : TComplex)');
18425:  Procedure CPower( A, B : TComplex; var Z : TComplex)');
18426:  Procedure CIntPower( A : TComplex; N : Integer; var Z : TComplex)');
18427:  Procedure CRealPower( A : TComplex; X : Float; var Z : TComplex)');
18428:  Procedure CSqrt( A : TComplex; var Z : TComplex)');
18429:  Procedure CRoot( A : TComplex; K, N : Integer; var Z : TComplex)');
18430:  Procedure CCSin( A : TComplex; var Z : TComplex)');
18431:  Procedure CCos( A : TComplex; var Z : TComplex)');
18432:  Procedure CTan( A : TComplex; var Z : TComplex)');
18433:  Procedure CArcSin( A : TComplex; var Z : TComplex)');
18434:  Procedure CArcCos( A : TComplex; var Z : TComplex)');
18435:  Procedure CArcTan( A : TComplex; var Z : TComplex)');
18436:  Procedure CSinh( A : TComplex; var Z : TComplex)');
18437:  Procedure CCosh( A : TComplex; var Z : TComplex)');
18438:  Procedure CTanh( A : TComplex; var Z : TComplex)');
18439:  Procedure CArcSinh( A : TComplex; var Z : TComplex)');
18440:  Procedure CArcCosh( A : TComplex; var Z : TComplex)');
18441:  Procedure CArcTanh( A : TComplex; var Z : TComplex)');
18442:  Procedure CLnGamma( A : TComplex; var Z : TComplex)');
18443: end;
18444:
18445:
18446: function TRestRequest_createStringStreamFromStringList(strings: TStringList): TStringStream;
18447:
18448: {A simple Oscilloscope using  TWaveIn class.
18449:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
18450:  http://www.retroarchive.org/garbo/pc/turbopas/index.html
18451: uses
18452:   Forms,
18453:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
18454:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
18455:   uColorFunctions in 'uColorFunctions.pas',
18456:   AMixer in 'AMixer.pas',
18457:   uSettings in 'uSettings.pas',
18458:   UWavein4 in 'UWavein4.pas',
18459:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
18460:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
18461:
18462: Functions_max hex in the box maXbox
18463: functionslist.txt
18464: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98/100/101/110/120/160
18465:
18466: ********************************************************************************
18467: Procedure
18468: PROCEDURE SIZE 8516 8396 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
18469: Procedure **************************Now the Procedure list******************
18470: Procedure ( ACol, ARow : Integer; Items : TStrings)
18471: Procedure ( Agg : TAggregate)
18472: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
18473: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
18474: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
18475: Procedure ( ASender : TObject; const ABytes : Integer)
18476: Procedure ( ASender : TObject; VStream : TStream)
18477: Procedure ( AThread : TIdThread)
18478: Procedure ( AWebModule : TComponent)
18479: Procedure ( Column : TColumn)
18480: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
18481: Procedure ( const iStart : integer; const sText : string)
18482: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
18483: Procedure ( Database : TDatabase; LoginParams : TStrings)
18484: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
18485: Procedure ( DATASET : TDATASET)
18486: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
```

```
18487: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
18488: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
18489: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
18490: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
18491: Procedure ( Done : Integer)
18492: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
18493: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
18494: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
18495: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
18496: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
18497: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
18498: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
18499: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
18500: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
18501: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
18502: Procedure ( SENDER : TFIELD; const TEXT : String)
18503: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
18504: Procedure ( Sender : TIdTelnet; const Buffer : String)
18505: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
18506: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
18507: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18508: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
18509: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
18510: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
18511: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
18512: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
18513: Procedure ( Sender : TObject; Button : TMPBtnType)
18514: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
18515: Procedure ( Sender : TObject; Button : TUDBtnType)
18516: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
18517: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
18518: Procedure ( Sender : TObject; Column : TListColumn)
18519: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
18520: Procedure ( Sender : TObject; Connecting : Boolean)
18521: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
18522: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
18523: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
18524: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
18525: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
18526: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
18527: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
18528: Procedure ( Sender : TObject; Index : LongInt)
18529: Procedure ( Sender : TObject; Item : TListItem)
18530: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
18531: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
18532: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
18533: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
18534: Procedure ( Sender : TObject; Item : TListItem; var S : string)
18535: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
18536: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
18537: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
18538: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
18539: Procedure ( Sender : TObject; Node : TTreeNode)
18540: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
18541: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
18542: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
18543: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
18544: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
18545: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
18546: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
18547: Procedure ( Sender : TObject; Rect : TRect)
18548: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
18549: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
18550: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
18551: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
18552: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
18553: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
18554: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
18555: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
18556: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
18557: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
18558: Procedure ( Sender : TObject; Thread : TServerClientThread)
18559: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
18560: Procedure ( Sender : TObject; Username, Password : string)
18561: Procedure ( Sender : TObject; var AllowChange : Boolean)
18562: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
18563: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
18564: Procedure ( Sender : TObject; var Continue : Boolean)
18565: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
18566: Procedure ( Sender : TObject; var Username : string)
18567: Procedure ( Sender : TObject; Wnd : HWND)
18568: Procedure ( Sender : TToolbar; Button : TToolButton)
18569: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
18570: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
18571: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
18572: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
18573: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
18574: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
```

```
18575: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
18576: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
18577: procedure (Sender: TObject)
18578: procedure (Sender: TObject; var Done: Boolean)
18579: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
18580: procedure _T(Name: tbtString; v: Variant);
18581: Procedure AbandonSignalHandler( RtlSigNum : Integer)
18582: Procedure Abort
18583: Procedure About1Click( Sender : TObject)
18584: Procedure Accept( Socket : TSocket)
18585: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
18586: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
18587: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
18588: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
18589: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
18590: Procedure Add( Addend1, Addend2 : TMyBigInt)
18591: Procedure ADD( const AKEY, AVALUE : VARIANT)
18592: Procedure Add( const Key : string; Value : Integer)
18593: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
18594: Procedure ADD( FIELD : TFIELD)
18595: Procedure ADD( ITEM : TMENUITEM)
18596: Procedure ADD( POPUP : TPOPUPMENU)
18597: Procedure AddCharacters( xCharacters : TCharSet)
18598: Procedure AddDriver( const Name : string; List : TStrings)
18599: Procedure AddImages( Value : TCustomImageList)
18600: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
18601: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
18602: Procedure AddLoader( Loader : TBitmapLoader)
18603: Procedure ADDPARAM( VALUE : TPARAM)
18604: Procedure AddPassword( const Password : string)
18605: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
18606: Procedure AddState( oState : TniRegularExpressionState)
18607: Procedure AddStrings( Strings : TStrings);
18608: procedure AddStrings(Strings: TStrings);
18609: Procedure AddStrings1( Strings : TWideStrings);
18610: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
18611: Procedure AddToRecentDocs( const Filename : string)
18612: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
18613: Procedure AllFunctionsList1Click( Sender : TObject)
18614: procedure AllObjectsList1Click(Sender: TObject);
18615: Procedure Allocate( AAllocateBytes : Integer)
18616: procedure AllResourceList1Click(Sender: TObject);
18617: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
18618: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
18619: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
18620: Procedure AnsiFree( var s : AnsiString)
18621: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
18622: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
18623: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
18624: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
18625: Procedure AntiFreeze;
18626: Procedure APPEND
18627: Procedure Append( const S : WideString)
18628: procedure Append(S: string);
18629: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
18630: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
18631: Procedure AppendChunk( Val : OleVariant)
18632: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
18633: Procedure AppendStr( var Dest : string; S : string)
18634: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
18635: Procedure ApplyRange
18636: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18637: Procedure Arrange( Code : TListArrangement)
18638: procedure Assert(expr : Boolean; const msg: string);
18639: procedure Assert2(expr : Boolean; const msg: string);
18640: Procedure Assign( AList : TCustomBucketList)
18641: Procedure Assign( Other : TObject)
18642: Procedure Assign( Source : TDragObject)
18643: Procedure Assign( Source : TPersistent)
18644: Procedure Assign(Source: TPersistent)
18645: procedure Assign2(mystring, mypath: string);
18646: Procedure AssignCurValues( Source : TDataSet);
18647: Procedure AssignCurValues1( const CurValues : Variant);
18648: Procedure ASSIGNFIELD( FIELD : TFIELD)
18649: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
18650: Procedure AssignFile(var F: Text; FileName: string)
18651: procedure AssignFile(var F: TextFile; FileName: string)
18652: procedure AssignFileRead(var mystring, myfilename: string);
18653: procedure AssignFileWrite(mystring, myfilename: string);
18654: Procedure AssignTo( Other : TObject)
18655: Procedure AssignValues( Value : TParameters)
18656: Procedure ASSIGNVALUES( VALUE : TPARAMS)
18657: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
18658: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
18659: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
18660: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
18661: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18662: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
18663: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
```

```
18664: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
18665: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
18666: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
18667: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
18668: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
18669: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
18670: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18671: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
18672: procedure Beep
18673: Procedure BeepOk
18674: Procedure BeepQuestion
18675: Procedure BeepHand
18676: Procedure BeepExclamation
18677: Procedure BeepAsterisk
18678: Procedure BeepInformation
18679: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
18680: Procedure BeginLayout
18681: Procedure BeginTimer( const Delay, Resolution : Cardinal)
18682: Procedure BeginUpdate
18683: procedure BeginUpdate;
18684: procedure BigScreen1Click(Sender: TObject);
18685: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
18686: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
18687: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
18688: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
18689: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
18690: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
18691: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
18692: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
18693: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
18694: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
18695: Procedure BreakPointMenuClick( Sender : TObject)
18696: procedure BRINGTOFRONT
18697: procedure BringToFront;
18698: Procedure btnBackClick( Sender : TObject)
18699: Procedure btnBrowseClick( Sender : TObject)
18700: Procedure BtnClick( Index : TNavigateBtn)
18701: Procedure btnLargeIconsClick( Sender : TObject)
18702: Procedure BuildAndSendRequest( AURI : TIdURI)
18703: Procedure BuildCache
18704: Procedure BurnMemory( var Buff, BuffLen : integer)
18705: Procedure BurnMemoryStream( Destructo : TMemoryStream)
18706: Procedure CalculateFirstSet
18707: Procedure Cancel
18708: procedure CancelDrag;
18709: Procedure CancelEdit
18710: procedure CANCELHINT
18711: Procedure CancelRange
18712: Procedure CancelUpdates
18713: Procedure CancelWriteBuffer
18714: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
18715: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
18716: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
18717: procedure CaptureScreenFormat(vname: string; vextension: string);
18718: procedure CaptureScreenPNG(vname: string);
18719: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
18720: procedure CASCADE
18721: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
18722: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
18723: Procedure cbPathClick( Sender : TObject)
18724: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18725: Procedure cedebugAfterExecute( Sender : TPSScript)
18726: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18727: Procedure cedebugCompile( Sender : TPSScript)
18728: Procedure cedebugExecute( Sender : TPSScript)
18729: Procedure cedebugIdle( Sender : TObject)
18730: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18731: Procedure CenterHeight( const pc, pcParent : TControl)
18732: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18733: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18734: Procedure Change
18735: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18736: Procedure Changed
18737: Procedure ChangeDir( const ADirName : string)
18738: Procedure ChangeDirUp
18739: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
18740: Procedure ChangeLevelBy( Value : TChangeRange)
18741: Procedure ChDir(const s: string)
18742: Procedure Check(Status: Integer)
18743: Procedure CheckCommonControl( CC : Integer)
18744: Procedure CHECKFIELDNAME( const FIELDNAME : String)
18745: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
18746: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
18747: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
18748: Procedure CheckToken( T : Char)
18749: procedure CheckToken(t:char)
18750: Procedure CheckTokenSymbol( const S : string)
18751: procedure CheckTokenSymbol(s:string)
18752: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
```

```
18753: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18754: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
18755: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
18756: procedure CipherFile1Click(Sender: TObject);
18757: Procedure Clear;
18758: Procedure Clear1Click( Sender : TObject)
18759: Procedure ClearColor( Color : TColor)
18760: Procedure CLEARITEM( AITEM : TMENUITEM)
18761: Procedure ClearMapping
18762: Procedure ClearSelection( KeepPrimary : Boolean)
18763: Procedure ClearWriteBuffer
18764: Procedure Click
18765: Procedure Close
18766: Procedure Close1Click( Sender : TObject)
18767: Procedure CloseDatabase( Database : TDatabase)
18768: Procedure CloseDataSets
18769: Procedure CloseDialog
18770: Procedure CloseFile(var F: Text);
18771: Procedure Closure
18772: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18773: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18774: Procedure CodeCompletionList1Click( Sender : TObject)
18775: Procedure ColEnter
18776: Procedure Collapse
18777: Procedure Collapse( Recurse : Boolean)
18778: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
18779: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
18780: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
18781: Procedure Compile1Click( Sender : TObject)
18782: procedure ComponentCount1Click(Sender: TObject);
18783: Procedure Compress(azipfolder, azipfile: string)
18784: Procedure DeCompress(azipfolder, azipfile: string)
18785: Procedure XZip(azipfolder, azipfile: string)
18786: Procedure XUnZip(azipfolder, azipfile: string)
18787: Procedure Connect( const ATimeout : Integer)
18788: Procedure Connect( Socket : TSocket)
18789: procedure Console1Click(Sender: TObject);
18790: Procedure Continue
18791: Procedure ContinueCount( var Counter : TJclCounter)
18792: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18793: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
18794: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
18795: Procedure ConvertImage(vsource, vdestination: string);
18796: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
18797: Procedure ConvertBitmap(vsource, vdestination: string);
18798: Procedure ConvertToGray(Cnv: TCanvas);
18799: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
18800: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
18801: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
18802: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
18803: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
18804: Procedure CopyFrom( mbCopy : TMyBigInt)
18805: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
18806: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
18807: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
18808: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
18809: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
18810: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
18811: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
18812: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
18813: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
18814: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18815: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
18816: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18817: Procedure CopyToClipboard
18818: Procedure CountParts
18819: Procedure CreateDataSet
18820: Procedure CreateEmptyFile( const FileName : string)
18821: Procedure CreateFileFromString( const FileName, Data : string)
18822: Procedure CreateFromDelta( Source : TPacketDataSet)
18823: procedure CREATEHANDLE
18824: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18825: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
18826: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
18827: Procedure CreateTable
18828: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
18829: procedure CSyntax1Click(Sender: TObject);
18830: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
18831: Procedure CURSORPOSCHANGED
18832: procedure CutFirstDirectory(var S: String)
18833: Procedure DataBaseError(const Message: string)
18834: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
18835: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
18836: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
18837: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18838: Procedure DBIError(errorCode: Integer)
```

```
18839: Procedure DebugOutput( const AText : string)
18840: procedure Debugln(DebugLOGFILE: string; Event_message: string);
18841: Procedure DebugRun1Click( Sender : TObject)
18842: procedure Dec;
18843: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
18844: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
18845: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
18846: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
18847: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
18848: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
18849: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18850: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
18851: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
18852: Procedure Decompile1Click( Sender : TObject)
18853: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18854: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18855: Procedure DeferLayout
18856: Procedure defFileread
18857: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
18858: Procedure DelayMicroseconds( const MicroSeconds : Integer)
18859: Procedure Delete
18860: Procedure Delete( const AFilename : string)
18861: Procedure Delete( const Index : Integer)
18862: Procedure DELETE( INDEX : INTEGER)
18863: Procedure Delete( Index : LongInt)
18864: Procedure Delete( Node : TTreeNode)
18865: procedure Delete(var s: AnyString; ifrom, icount: Longint);
18866: Procedure DeleteAlias( const Name : string)
18867: Procedure DeleteDriver( const Name : string)
18868: Procedure DeleteIndex( const Name : string)
18869: Procedure DeleteKey( const Section, Ident : String)
18870: Procedure DeleteRecords
18871: Procedure DeleteRecords( AffectRecords : TAffectRecords)
18872: Procedure DeleteString( var pStr : String; const pDelStr : string)
18873: Procedure DeleteTable
18874: procedure DelphiSite1Click(Sender: TObject);
18875: Procedure Deselect
18876: Procedure Deselect( Node : TTreeNode)
18877: procedure DestroyComponents
18878: Procedure DestroyHandle
18879: Procedure Diff( var X : array of Double)
18880: procedure Diff(var X: array of Double);
18881: Procedure DirCreate( const DirectoryName : String)');
18882: procedure DISABLEALIGN
18883: Procedure DisableConstraints
18884: Procedure Disconnect
18885: Procedure Disconnect( Socket : TSocket)
18886: Procedure Dispose
18887: procedure Dispose(P: PChar)
18888: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
18889: Procedure DoKey( Key : TDBCtrlGridKey)
18890: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18891: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18892: Procedure Dormant
18893: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
18894: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
18895: Procedure DoubleToComp( Value : Double; var Result : Comp)
18896: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18897: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
18898: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18899: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18900: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18901: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
18902: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18903: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
18904: procedure DrawFocusRect(const Rect: TRect);
18905: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
18906: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18907: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
18908: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
18909: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18910: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
18911: Procedure DropConnections
18912: Procedure DropDown
18913: Procedure DumpDescription( oStrings : TStrings)
18914: Procedure DumpStateTable( oStrings : TStrings)
18915: Procedure EDIT
18916: Procedure EditButtonClick
18917: Procedure EditFont1Click( Sender : TObject)
18918: procedure Ellipse(X1, Y1, X2, Y2: Integer);
18919: Procedure Ellipse1( const Rect : TRect);
18920: Procedure EMMS
18921: Procedure Encode( ADest : TStream)
18922: procedure ENDDRAG(DROP:BOOLEAN)
18923: Procedure EndEdit( Cancel : Boolean)
18924: Procedure EndTimer
18925: Procedure EndUpdate
```

```
18926: Procedure EraseSection( const Section : string)
18927: Procedure Error( const Ident : string)
18928: procedure Error(Ident:Integer)
18929: Procedure ErrorFmt( const Ident : string; const Args : array of const)
18930: Procedure ErrorStr( const Message : string)
18931: procedure ErrorStr(Message:String)
18932: Procedure Exchange( Index1, Index2 : Integer)
18933: procedure Exchange(Index1, Index2: Integer);
18934: Procedure Exec( FileName, Parameters, Directory : string)
18935: Procedure ExecProc
18936: Procedure ExecSQL( UpdateKind : TUpdateKind)
18937: Procedure Execute
18938: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
18939: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
18940: Procedure ExecuteCommand(executeFile, paramstring: string)
18941: Procedure ExecuteShell(executeFile, paramstring: string)
18942: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18943: Procedure ExitThread(ExitCode: Integer); stdcall;
18944: Procedure ExitProcess(ExitCode: Integer); stdcall;
18945: Procedure Expand( AUserName : String; AResults : TStrings)
18946: Procedure Expand( Recurse : Boolean)
18947: Procedure ExportClipboard1Click( Sender : TObject)
18948: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
18949: Procedure ExtractContentFields( Strings : TStrings)
18950: Procedure ExtractCookieFields( Strings : TStrings)
18951: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
18952: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18953: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18954: Procedure ExtractQueryFields( Strings : TStrings)
18955: Procedure FastDegToGrad
18956: Procedure FastDegToRad
18957: Procedure FastGradToDeg
18958: Procedure FastGradToRad
18959: Procedure FastRadToDeg
18960: Procedure FastRadToGrad
18961: Procedure FileClose( Handle : Integer)
18962: Procedure FileClose(handle: integer)
18963: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18964: Procedure FileStructure( AStructure : TIdFTPDataStructure)
18965: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
18966: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
18967: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
18968: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18969: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18970: Procedure FillIPList
18971: procedure FillRect(const Rect: TRect);
18972: Procedure FillTStrings( AStrings : TStrings)
18973: Procedure FilterOnBookmarks( Bookmarks : array of const)
18974: procedure FinalizePackage(Module: HMODULE)
18975: procedure FindClose;
18976: procedure FindClose2(var F: TSearchRec)
18977: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
18978: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
18979: Procedure FindNearest( const KeyValues : array of const)
18980: Procedure FinishContext
18981: Procedure FIRST
18982: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
18983: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18984: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
18985: Procedure FlushSchemaCache( const TableName : string)
18986: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
18987: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
18988: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
18989: Procedure FormActivate( Sender : TObject)
18990: procedure FormatLn(const format: String; const args: array of const);  //alias
18991: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18992: Procedure FormCreate( Sender : TObject)
18993: Procedure FormDestroy( Sender : TObject)
18994: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18995: procedure FormOutput1Click(Sender: TObject);
18996: Procedure FormToHtml( Form : TForm; Path : string)
18997: procedure FrameRect(const Rect: TRect);
18998: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18999: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
19000: Procedure Free( Buffer : TRecordBuffer)
19001: Procedure Free( Buffer : TValueBuffer)
19002: Procedure Free;
19003: Procedure FreeAndNil(var Obj:TObject)
19004: Procedure FreeImage
19005: procedure FreeMem(P: PChar; Size: Integer)
19006: Procedure FreeTreeData( Tree : TUpdateTree)
19007: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
19008: Procedure FullCollapse
19009: Procedure FullExpand
19010: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
19011: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
19012: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
```

```
19013: Procedure Get1( AURL : string; const AResponseContent : TStream);
19014: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
19015: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
19016: Procedure GetAliasNames( List : TStrings)
19017: Procedure GetAliasParams( const AliasName : string; List : TStrings)
19018: Procedure GetApplicationsRunning( Strings : TStrings)
19019: Procedure getBox(aURL, extension: string);
19020: Procedure GetCommandTypes( List : TWideStrings)
19021: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
19022: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
19023: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
19024: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
19025: Procedure GetDatabaseNames( List : TStrings)
19026: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
19027: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
19028: Procedure GetDir(d: byte; var s: string)
19029: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
19030: Procedure GetDriverNames( List : TStrings)
19031: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
19032: Procedure GetDriverParams( const DriverName : string; List : TStrings)
19033: Procedure GetEMails1Click( Sender : TObject)
19034: Procedure getEnvironmentInfo;
19035: Function getEnvironmentString: string;
19036: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
19037: Procedure GetFieldNames( const TableName : string; List : TStrings)
19038: Procedure GetFieldNames( const TableName : string; List : TStrings);
19039: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
19040: Procedure GETFIELDNAMES( LIST : TSTRINGS)
19041: Procedure GetFieldNames1( const TableName : string; List : TStrings);
19042: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
19043: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
19044: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
19045: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
19046: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
19047: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
19048: Procedure GetFormatSettings
19049: Procedure GetFromDIB( var DIB : TBitmapInfo)
19050: Procedure GetFromHDIB( HDIB : HBitmap)
19051: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
19052: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
19053: Procedure GetIcon( Index : Integer; Image : TIcon);
19054: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
19055: Procedure GetIndexInfo( IndexName : string)
19056: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
19057: Procedure GetIndexNames( List : TStrings)
19058: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
19059: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
19060: Procedure GetIndexNames4( const TableName : string; List : TStrings);
19061: Procedure GetInternalResponse
19062: Procedure GETITEMNAMES( LIST : TSTRINGS)
19063: procedure GetMem(P: PChar; Size: Integer)
19064: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
19065: procedure GetPackageDescription(ModuleName: PChar): string)
19066: Procedure GetPackageNames( List : TStrings);
19067: Procedure GetPackageNames1( List : TWideStrings);
19068: Procedure GetParamList( List : TList; const ParamNames : WideString)
19069: Procedure GetProcedureNames( List : TStrings);
19070: Procedure GetProcedureNames( List : TWideStrings);
19071: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
19072: Procedure GetProcedureNames1( List : TStrings);
19073: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
19074: Procedure GetProcedureNames3( List : TWideStrings);
19075: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
19076: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
19077: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
19078: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
19079: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
19080: Procedure GetProviderNames( Names : TWideStrings);
19081: Procedure GetProviderNames( Proc : TGetStrProc)
19082: Procedure GetProviderNames1( Names : TStrings);
19083: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
19084: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
19085: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
19086: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
19087: Procedure GetSchemaNames( List : TStrings);
19088: Procedure GetSchemaNames1( List : TWideStrings);
19089: Procedure getScriptandRunAsk;
19090: Procedure getScriptandRun(ascript: string);
19091: Procedure getScript(ascript: string);   //alias
19092: Procedure getWebScript(ascript: string);   //alias
19093: Procedure GetSessionNames( List : TStrings)
19094: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
19095: Procedure GetStrings( List : TStrings)
19096: Procedure GetSystemTime; stdcall;
19097: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
19098: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
19099: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
19100: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
```

```
19101: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
19102: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
19103: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
19104: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
19105: Procedure GetVisibleWindows( List : Tstrings)
19106: Procedure GoBegin
19107: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
19108: Procedure GotoCurrent( Table : TTable)
19109: procedure GotoEnd1Click(Sender: TObject);
19110: Procedure GotoNearest
19111: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
19112: Procedure HandleException( E : Exception; var Handled : Boolean)
19113: procedure HANDLEMESSAGE
19114: procedure HandleNeeded;
19115: Procedure Head( AURL : string)
19116: Procedure Help( var AHelpContents : TStringList; ACommand : String)
19117: Procedure HexToBinary( Stream : TStream)
19118: procedure HexToBinary(Stream:TStream)
19119: Procedure HideDragImage
19120: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
19121: Procedure HideTraybar
19122: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
19123: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
19124: Procedure HookOSExceptions
19125: Procedure HookSignal( RtlSigNum : Integer)
19126: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
19127: Procedure HTMLSyntax1Click( Sender : TObject)
19128: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
19129: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
19130: Procedure ImportfromClipboard1Click( Sender : TObject)
19131: Procedure ImportfromClipboard2Click( Sender : TObject)
19132: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
19133: procedure Incb(var x: byte);
19134: Procedure Include1Click( Sender : TObject)
19135: Procedure IncludeOFF;  //preprocessing
19136: Procedure IncludeON;
19137: procedure Info1Click(Sender: TObject);
19138: Procedure InitAltRecBuffers( CheckModified : Boolean)
19139: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
19140: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
19141: Procedure InitData( ASource : TDataSet)
19142: Procedure InitDelta( ADelta : TPacketDataSet);
19143: Procedure InitDelta1( const ADelta : OleVariant);
19144: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
19145: Procedure Initialize
19146: procedure InitializePackage(Module: HMODULE)
19147: Procedure INITIATEACTION
19148: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
19149: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
19150: Procedure InitModule( AModule : TComponent)
19151: Procedure InitStdConvs
19152: Procedure InitTreeData( Tree : TUpdateTree)
19153: Procedure INSERT
19154: Procedure Insert( Index : Integer; AClass : TClass)
19155: Procedure Insert( Index : Integer; AComponent : TComponent)
19156: Procedure Insert( Index : Integer; AObject : TObject)
19157: Procedure Insert( Index : Integer; const S : WideString)
19158: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
19159: Procedure Insert(Index: Integer; const S: string);
19160: procedure Insert(Index: Integer; S: string);
19161: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
19162: procedure InsertComponent(AComponent:TComponent)
19163: procedure InsertControl(AControl: TControl);
19164: Procedure InsertIcon( Index : Integer; Image : TIcon)
19165: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
19166: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
19167: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
19168: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
19169: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
19170: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
19171: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
19172: Procedure InternalBeforeResolve( Tree : TUpdateTree)
19173: Procedure InvalidateModuleCache
19174: Procedure InvalidateTitles
19175: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
19176: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
19177: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
19178: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
19179: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
19180: procedure JavaSyntax1Click(Sender: TObject);
19181: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
19182: Procedure KillDataChannel
19183: Procedure Largefont1Click( Sender : TObject)
19184: Procedure LAST
19185: Procedure LaunchCpl( FileName : string)
19186: Procedure Launch( const AFile : string)
19187: Procedure LaunchFile( const AFile : string)
```

```
19188: Procedure LetFileList(FileList: TStringlist; apath: string);
19189: Procedure lineToNumber( xmemo : String; met : boolean)
19190: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
19191: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
19192: Procedure ListViewData( Sender : TObject; Item : TListItem)
19193: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
19194: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
19195: Procedure ListViewDblClick( Sender : TObject)
19196: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
19197: Procedure ListDLLExports(const FileName: string; List: TStrings);
19198: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
19199: procedure LoadBytecode1Click(Sender: TObject);
19200: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
19201: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
19202: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
19203: Procedure LoadFromFile( AFileName : string)
19204: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
19205: Procedure LoadFromFile( const FileName : string)
19206: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
19207: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
19208: Procedure LoadFromFile( const FileName : WideString)
19209: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19210: Procedure LoadFromFile(const AFileName: string)
19211: procedure LoadFromFile(FileName: string);
19212: procedure LoadFromFile(FileName:String)
19213: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
19214: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
19215: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
19216: Procedure LoadFromStream( const Stream : TStream)
19217: Procedure LoadFromStream( S : TStream)
19218: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
19219: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19220: Procedure LoadFromStream( Stream : TStream)
19221: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
19222: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
19223: procedure LoadFromStream(Stream: TStream);
19224: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
19225: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
19226: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
19227: Procedure LoadLastFile1Click( Sender : TObject)
19228: { LoadIcoToImage loads two icons from resource named NameRes,
19229:   into two image lists ALarge and ASmall}
19230: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
19231: Procedure LoadMemo
19232: Procedure LoadParamsFromIniFile( FFileName : WideString)
19233: Procedure Lock
19234: Procedure Login
19235: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
19236: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
19237: Procedure MakeCaseInsensitive
19238: Procedure MakeDeterministic( var bChanged : boolean)
19239: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
19240: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
19241: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
19242: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
19243: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19244: Procedure SetRectComplexFormatStr( const S : string)
19245: Procedure SetPolarComplexFormatStr( const S : string)
19246: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19247: Procedure MakeVisible
19248: Procedure MakeVisible( PartialOK : Boolean)
19249: Procedure Manual1Click( Sender : TObject)
19250: Procedure MarkReachable
19251: Procedure maXbox;  //shows the exe version data in a win box
19252: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
19253: Procedure Memo1Change( Sender : TObject)
19254: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
19255: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
19256: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
19257: procedure Memory1Click(Sender: TObject);
19258: Procedure MERGE( MENU : TMAINMENU)
19259: Procedure MergeChangeLog
19260: procedure MINIMIZE
19261: Procedure MinimizeMaxbox;
19262: procedure MyCopyFile(Name1,Name2:string);
19263: Procedure MkDir(const s: string)
19264: Procedure MakeDir(const s: string)');
19265: Procedure ChangeDir(const s: string)');
19266: Function makeFile(const FileName: string): integer)');
19267: Procedure mnuPrintFont1Click( Sender : TObject)
19268: procedure ModalStarted
19269: Procedure Modified
19270: Procedure ModifyAlias( Name : string; List : TStrings)
19271: Procedure ModifyDriver( Name : string; List : TStrings)
19272: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
```

```
19273: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
19274: Procedure Move( CurIndex, NewIndex : Integer)
19275: procedure Move(CurIndex, NewIndex: Integer);
19276: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
19277: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
19278: Procedure moveCube( o : TMyLabel)
19279: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
19280: procedure MoveTo(X, Y: Integer);
19281: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
19282: Procedure MovePoint(var x,y:Extended; const angle:Extended);
19283: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
19284: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
19285: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
19286: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
19287: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
19288: procedure New(P: PChar)
19289: procedure New1Click(Sender: TObject);
19290: procedure NewInstance1Click(Sender: TObject);
19291: Procedure NEXT
19292: Procedure NextMonth
19293: Procedure Noop
19294: Procedure NormalizePath( var APath : string)
19295: procedure ObjectBinaryToText(Input, Output: TStream)
19296: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19297: procedure ObjectResourceToText(Input, Output: TStream)
19298: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19299: procedure ObjectTextToBinary(Input, Output: TStream)
19300: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19301: procedure ObjectTextToResource(Input, Output: TStream)
19302: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
19303: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
19304: Procedure Open( const UserID : WideString; const Password : WideString);
19305: Procedure Open;
19306: Procedure open1Click( Sender : TObject)
19307: Procedure OpenCdDrive
19308: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
19309: Procedure OpenCurrent
19310: Procedure OpenFile(vfilenamepath: string)
19311: Procedure OpenDirectory1Click( Sender : TObject)
19312: Procedure OpenDir(adir: string);
19313: Procedure OpenIndexFile( const IndexName : string)
19314: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
19315: Procedure OpenWriteBuffer( const AThreshhold : Integer)
19316: Procedure OptimizeMem
19317: Procedure Options1( AURL : string);
19318: Procedure OutputDebugString(lpOutputString : PChar)
19319: Procedure PackBuffer
19320: Procedure Paint
19321: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
19322: Procedure PaintToTBitmap( Target : TBitmap)
19323: Procedure PaletteChanged
19324: Procedure ParentBiDiModeChanged
19325: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
19326: Procedure PasteFromClipboard;
19327: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
19328: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
19329: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
19330: Procedure PError( Text : string)
19331: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
19332: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
19333: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
19334: procedure playmp3(mpath: string);
19335: Procedure PlayMP31Click( Sender : TObject)
19336: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
19337: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
19338: procedure PolyBezier(const Points: array of TPoint);
19339: procedure PolyBezierTo(const Points: array of TPoint);
19340: procedure Polygon(const Points: array of TPoint);
19341: procedure Polyline(const Points: array of TPoint);
19342: Procedure Pop
19343: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
19344: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
19345: Procedure POPUP( X, Y : INTEGER)
19346: Procedure PopupURL(URL : WideString);
19347: Procedure POST
19348: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
19349: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
19350: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
19351: Procedure PostUser( const Email, FirstName, LastName : WideString)
19352: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19353: procedure Pred(X: int64);
19354: Procedure Prepare
19355: Procedure PrepareStatement
19356: Procedure PreProcessXML( AList : TStrings)
19357: Procedure PreventDestruction
19358: Procedure Print( const Caption : string)
19359: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
19360: procedure printf(const format: String; const args: array of const);
```

```
19361: Procedure PrintList(Value: TStringList);
19362: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
19363: Procedure Printout1Click( Sender : TObject)
19364: Procedure ProcessHeaders
19365: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
19366: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
19367: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
19368: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
19369: Procedure ProcessMessagesOFF;  //application.processmessages
19370: Procedure ProcessMessagesON;
19371: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
19372: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
19373: Procedure Proclist Size is: 3797 /1415
19374: Procedure procMessClick( Sender : TObject)
19375: Procedure PSScriptCompile( Sender : TPSScript)
19376: Procedure PSScriptExecute( Sender : TPSScript)
19377: Procedure PSScriptLine( Sender : TObject)
19378: Procedure Push( ABoundary : string)
19379: procedure PushItem(AItem: Pointer)
19380: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
19381: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
19382: procedure PutLinuxLines(const Value: string)
19383: Procedure Quit
19384: Procedure RaiseConversionError( const AText : string);
19385: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
19386: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
19387: procedure RaiseException(Ex: TIFException; Param: String);
19388: Procedure RaiseExceptionForLastCmdResult;
19389: procedure RaiseLastException;
19390: procedure RaiseException2;
19391: Procedure RaiseException3(const Msg: string);
19392: Procedure RaiseExcept(const Msg: string);
19393: Procedure RaiseLastOSError
19394: Procedure RaiseLastWin32;
19395: procedure RaiseLastWin32Error)
19396: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
19397: Procedure RandomFillStream( Stream : TMemoryStream)
19398: procedure randomize;
19399: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
19400: Procedure RCS
19401: Procedure Read( Socket : TSocket)
19402: procedure Readln1(var ast: string);  //of inputquery
19403: Procedure ReadBlobData
19404: procedure ReadBuffer(Buffer:String;Count:LongInt)
19405: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
19406: Procedure ReadSection( const Section : string; Strings : TStrings)
19407: Procedure ReadSections( Strings : TStrings)
19408: Procedure ReadSections( Strings : TStrings);
19409: Procedure ReadSections1( const Section : string; Strings : TStrings);
19410: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
19411: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
19412: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
19413: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
19414: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
19415: Procedure Realign;
19416: procedure Rectangle(X1, Y1, X2, Y2: Integer);
19417: Procedure Rectangle1( const Rect : TRect);
19418: Procedure RectCopy( var Dest : TRect; const Source : TRect)
19419: Procedure RectFitToScreen( var R : TRect)
19420: Procedure RectGrow( var R : TRect; const Delta : Integer)
19421: Procedure RectGrowX( var R : TRect; const Delta : Integer)
19422: Procedure RectGrowY( var R : TRect; const Delta : Integer)
19423: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
19424: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
19425: Procedure RectNormalize( var R : TRect)
19426: //  TFileCallbackProcedure = procedure(filename:string);
19427: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
19428: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
19429: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
19430: Procedure Refresh;
19431: Procedure RefreshData( Options : TFetchOptions)
19432: Procedure REFRESHLOOKUPLIST
19433: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
19434: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
19435: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
19436: Procedure RegisterChanges( Value : TChangeLink)
19437: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
19438: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
19439: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
19440: Procedure ReInitialize( ADelay : Cardinal)
19441: procedure RELEASE
19442: Procedure Remove( const AByteCount : integer)
19443: Procedure REMOVE( FIELD : TFIELD)
19444: Procedure REMOVE( ITEM : TMENUITEM)
19445: Procedure REMOVE( POPUP : TPOPUPMENU)
19446: Procedure RemoveAllPasswords
19447: procedure RemoveComponent(AComponent:TComponent)
19448: Procedure RemoveDir( const ADirName : string)
19449: Procedure RemoveLambdaTransitions( var bChanged : boolean)
```

```
19450: Procedure REMOVEPARAM( VALUE : TPARAM)
19451: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
19452: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
19453: Procedure Rename( const ASourceFile, ADestFile : string)
19454: Procedure Rename( const FileName : string; Reload : Boolean)
19455: Procedure RenameTable( const NewTableName : string)
19456: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
19457: Procedure Replace1Click( Sender : TObject)
19458: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
19459: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
19460: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
19461: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
19462: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
19463: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
19464: Procedure Requery( Options : TExecuteOptions)
19465: Procedure Reset
19466: Procedure Reset1Click( Sender : TObject)
19467: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
19468: procedure ResourceExplore1Click(Sender: TObject);
19469: Procedure RestoreContents
19470: Procedure RestoreDefaults
19471: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
19472: Procedure RetrieveHeaders
19473: Procedure RevertRecord
19474: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
19475: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19476: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19477: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
19478: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
19479: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
19480: Procedure RleCompress2( Stream : TStream)
19481: Procedure RleDecompress2( Stream : TStream)
19482: Procedure RmDir(const S: string)
19483: Procedure Rollback
19484: Procedure Rollback( TransDesc : TTransactionDesc)
19485: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
19486: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
19487: Procedure RollbackTrans
19488: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
19489: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
19490: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
19491: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
19492: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
19493: Procedure S_EBox( const AText : string)
19494: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
19495: Procedure S_IBox( const AText : string)
19496: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
19497: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
19498: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
19499: Procedure SampleVarianceAndMean
19500: ( const X : TDynFloatArray; var Variance, Mean : Float)
19501: Procedure Save2Click( Sender : TObject)
19502: Procedure Saveas3Click( Sender : TObject)
19503: Procedure Savebefore1Click( Sender : TObject)
19504: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
19505: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19506: Procedure SaveConfigFile
19507: Procedure SaveOutput1Click( Sender : TObject)
19508: procedure SaveScreenshotClick(Sender: TObject);
19509: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
19510: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
19511: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
19512: Procedure SaveToFile( AFileName : string)
19513: Procedure SAVETOFILE( const FILENAME : String)
19514: Procedure SaveToFile( const FileName : WideString)
19515: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
19516: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19517: procedure SaveToFile(FileName: string);
19518: procedure SaveToFile(FileName:String)
19519: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
19520: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
19521: Procedure SaveToStream( S : TStream)
19522: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19523: Procedure SaveToStream( Stream : TStream)
19524: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
19525: procedure SaveToStream(Stream: TStream);
19526: procedure SaveToStream(Stream:TStream)
19527: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
19528: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
19529: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
19530: procedure Say(const sText: string)
19531: Procedure SBytecode1Click( Sender : TObject)
19532: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
19533: procedure ScriptExplorer1Click(Sender: TObject);
19534: Procedure Scroll( Distance : Integer)
19535: Procedure Scroll( DX, DY : Integer)
19536: procedure ScrollBy(DeltaX, DeltaY: Integer);
19537: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
```

```
19538: Procedure ScrollTabs( Delta : Integer)
19539: Procedure Search1Click( Sender : TObject)
19540: procedure SearchAndOpenDoc(vfilenamepath: string)
19541: procedure SearchAndOpenFile(vfilenamepath: string)
19542: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
19543: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19544: Procedure SearchNext1Click( Sender : TObject)
19545: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
19546: Procedure Select1( const Nodes : array of TTreeNode);
19547: Procedure Select2( Nodes : TList);
19548: Procedure SelectNext( Direction : Boolean)
19549: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
19550: Procedure SelfTestPEM  //unit uPSI_cPEM
19551: Procedure Send( AMsg : TIdMessage)
19552: //config forst in const MAILINIFILE = 'maildef.ini';
19553: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
19554: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19555: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19556: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
19557: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
19558: Procedure SendResponse
19559: Procedure SendStream( AStream : TStream)
19560: procedure SendMCICommand(Cmd: string);   !
19561: Procedure Set8087CW( NewCW : Word)
19562: Procedure SetAll( One, Two, Three, Four : Byte)
19563: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
19564: Procedure SetAppDispatcher( const ADispatcher : TComponent)
19565: procedure SetArrayLength;
19566: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
19567: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19568: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer);  //all init
19569: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
19570: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);');
19571: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);');
19572: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
19573: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
19574: Procedure SetAsHandle( Format : Word; Value : THandle)
19575: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
19576: procedure SetCaptureControl(Control: TControl);
19577: Procedure SetColumnAttributes
19578: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
19579: Procedure SetCustomHeader( const Name, Value : string)
19580: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
     FieldName:Widestring)
19581: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
19582: Procedure SetFocus
19583: procedure SetFocus; virtual;
19584: Procedure SetInitialState
19585: Procedure SetKey
19586: procedure SetLastError(ErrorCode: Integer)
19587: procedure SetLength;
19588: procedure SetLength2(var S: string; NewLength: Integer)');
19589: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
19590: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
19591: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
19592: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
19593: Procedure SetParams1( UpdateKind : TUpdateKind);
19594: Procedure SetPassword( const Password : string)
19595: Procedure SetPointer( Ptr : Pointer; Size : Longint)
19596: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
19597: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
19598: Procedure SetProvider( Provider : TComponent)
19599: Procedure SetProxy( const Proxy : string)
19600: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
19601: Procedure SetRange( const StartValues, EndValues : array of const)
19602: Procedure SetRangeEnd
19603: Procedure SetRate( const aPercent, aYear : integer)
19604: procedure SetRate(const aPercent, aYear: integer)
19605: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19606: Procedure SetSafeCallExceptionMsg( Msg : String)
19607: procedure SETSELTEXTBUF(BUFFER:PCHAR)
19608: Procedure SetSize( AWidth, AHeight : Integer)
19609: procedure SetSize(NewSize:LongInt)
19610: procedure SetString(var s: string; buffer: PChar; len: Integer)
19611: Procedure SetStrings( List : TStrings)
19612: Procedure SetText( Text : PwideChar)
19613: procedure SetText(Text: PChar);
19614: Procedure SetTextBuf( Buffer : PChar)
19615: procedure SETTEXTBUF(BUFFER:PCHAR)
19616: Procedure SetTick( Value : Integer)
19617: Procedure SetTimeout( ATimeOut : Integer)
19618: Procedure SetTraceEvent( Event : TDBXTraceEvent)
19619: Procedure SetUserName( const UserName : string)
19620: Procedure SetWallpaper( Path : string);
19621: procedure ShellStyle1Click(Sender: TObject);
19622: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
19623: Procedure ShowFileProperties( const FileName : string)
19624: Procedure ShowInclude1Click( Sender : TObject)
19625: Procedure ShowInterfaces1Click( Sender : TObject)
```

```
19626: Procedure ShowLastException1Click( Sender : TObject)
19627: Procedure ShowMessage( const Msg : string)
19628: Procedure ShowMessageBig(const aText : string);
19629: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
19630: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
19631: Procedure MsgBig(const aText : string);        //alias
19632: procedure showmessage(mytext: string);
19633: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
19634: procedure ShowMessageFmt(const Msg: string; Params: array of const))
19635: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
19636: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
19637: Procedure ShowSearchDialog( const Directory : string)
19638: Procedure ShowSpecChars1Click( Sender : TObject)
19639: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
19640: Procedure ShredFile( const FileName : string; Times : Integer)
19641: procedure Shuffle(vQ: TStringList);
19642: Procedure ShuffleList( var List : array of Integer; Count : Integer)
19643: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
19644: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
19645: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
19646: Procedure Site( const ACommand : string)
19647: Procedure SkipEOL
19648: Procedure Sleep( ATime : cardinal)
19649: Procedure Sleep( milliseconds : Cardinal)
19650: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
19651: Procedure Slinenumbers1Click( Sender : TObject)
19652: Procedure Sort
19653: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
19654: procedure SoundAlarm;  //beep seq
19655: procedure Speak(const sText: string)   //async like voice
19656: procedure Speak2(const sText: string)  //sync
19657: procedure Split(Str: string;  SubStr: string; List: TStrings);
19658: Procedure SplitNameValue( const Line : string; var Name, Value : string)
19659: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
19660: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
19661: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
19662: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
19663: procedure SQLSyntax1Click(Sender: TObject);
19664: Procedure SRand( Seed : RNG_IntType)
19665: Procedure Start
19666: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
19667: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
19668: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
19669: Procedure StartTransaction( TransDesc : TTransactionDesc)
19670: Procedure Status( var AStatusList : TStringList)
19671: Procedure StatusBar1DblClick( Sender : TObject)
19672: Procedure StepInto1Click( Sender : TObject)
19673: Procedure StepIt
19674: Procedure StepOut1Click( Sender : TObject)
19675: Procedure Stop
19676: procedure stopmp3;
19677: procedure StartWeb(aurl: string);
19678: Procedure Str(aint: integer; astr: string); //of system
19679: Procedure StrDispose( Str : PChar)
19680: procedure StrDispose(Str: PChar)
19681: Procedure StrReplace(var Str: String; Old, New: String);
19682: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
19683: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
19684: Procedure StringToBytes( Value : String; Bytes : array of byte)
19685: procedure StrSet(c : Char; I : Integer; var s : String);
19686: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19687: Procedure StructureMount( APath : String)
19688: procedure STYLECHANGED(SENDER:TOBJECT)
19689: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
19690: procedure Succ(X: int64);
19691: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
19692: procedure SwapChar(var X,Y: char);  //swapX follows
19693: Procedure SwapFloats( var X, Y : Float)
19694: procedure SwapGrid(grd: TStringGrid);
19695: Procedure SwapOrd( var I, J : Byte);
19696: Procedure SwapOrd( var X, Y : Integer)
19697: Procedure SwapOrd1( var I, J : Shortint);
19698: Procedure SwapOrd2( var I, J : Smallint);
19699: Procedure SwapOrd3( var I, J : Word);
19700: Procedure SwapOrd4( var I, J : Integer);
19701: Procedure SwapOrd5( var I, J : Cardinal);
19702: Procedure SwapOrd6( var I, J : Int64);
19703: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
19704: Procedure Synchronize1( Method : TMethod);
19705: procedure SyntaxCheck1Click(Sender: TObject);
19706: Procedure SysFreeString(const S: WideString); stdcall;
19707: Procedure TakeOver( Other : TLinearBitmap)
19708: Procedure Talkln(const sText: string)  //async voice
19709: procedure tbtn6resClick(Sender: TObject);
19710: Procedure tbtnUseCaseClick( Sender : TObject)
19711: procedure TerminalStyle1Click(Sender: TObject);
19712: Procedure Terminate
19713: Procedure texSyntax1Click( Sender : TObject)
19714: procedure TextOut(X, Y: Integer; Text: string);
```

```
19715: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
19716: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
19717: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
19718: Procedure TextStart
19719: procedure TILE
19720: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
19721: Procedure TitleClick( Column : TColumn)
19722: Procedure ToDo
19723: procedure toolbtnTutorialClick(Sender: TObject);
19724: Procedure Trace1( AURL : string; const AResponseContent : TStream);
19725: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
19726: Procedure Truncate
19727: procedure Tutorial101Click(Sender: TObject);
19728: procedure Tutorial10Statistics1Click(Sender: TObject);
19729: procedure Tutorial11Forms1Click(Sender: TObject);
19730: procedure Tutorial12SQL1Click(Sender: TObject);
19731: Procedure tutorial1Click( Sender : TObject)
19732: Procedure tutorial21Click( Sender : TObject)
19733: Procedure tutorial31Click( Sender : TObject)
19734: Procedure tutorial4Click( Sender : TObject)
19735: Procedure Tutorial5Click( Sender : TObject)
19736: procedure Tutorial6Click(Sender: TObject);
19737: procedure Tutorial91Click(Sender: TObject);
19738: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19739: procedure UniqueString(var str: AnsiString)
19740: procedure UnloadLoadPackage(Module: HMODULE)
19741: Procedure Unlock
19742: Procedure UNMERGE( MENU : TMAINMENU)
19743: Procedure UnRegisterChanges( Value : TChangeLink)
19744: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19745: Procedure UnregisterConversionType( const AType : TConvType)
19746: Procedure UnRegisterProvider( Prov : TCustomProvider)
19747: Procedure UPDATE
19748: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19749: Procedure UPDATECURSORPOS
19750: Procedure UpdateFile
19751: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19752: Procedure UpdateResponse( AResponse : TWebResponse)
19753: Procedure UpdateScrollBar
19754: Procedure UpdateView1Click( Sender : TObject)
19755: procedure UpdateExeResource(Const Source,Dest:string);  //!
19756: procedure Val(const s: string; var n, z: Integer)
19757: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19758: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
19759: Procedure VariantAdd( const src : Variant; var dst : Variant)
19760: Procedure VariantAnd( const src : Variant; var dst : Variant)
19761: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19762: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19763: Procedure VariantClear( var V : Variant)
19764: Procedure VariantCpy( const src : Variant; var dst : Variant)
19765: Procedure VariantDiv( const src : Variant; var dst : Variant)
19766: Procedure VariantMod( const src : Variant; var dst : Variant)
19767: Procedure VariantMul( const src : Variant; var dst : Variant)
19768: Procedure VariantOr( const src : Variant; var dst : Variant)
19769: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
19770: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
19771: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
19772: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
19773: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
19774: Procedure VariantShl( const src : Variant; var dst : Variant)
19775: Procedure VariantShr( const src : Variant; var dst : Variant)
19776: Procedure VariantSub( const src : Variant; var dst : Variant)
19777: Procedure VariantXor( const src : Variant; var dst : Variant)
19778: Procedure VarCastError;
19779: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19780: Procedure VarInvalidOp
19781: Procedure VarInvalidNullOp
19782: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19783: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19784: Procedure VarArrayCreateError
19785: Procedure VarResultCheck( AResult : HRESULT);
19786: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19787: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19788: Function VarTypeAsText( const AType : TVarType) : string
19789: procedure Voice(const sText: string) //async
19790: procedure Voice2(const sText: string) //sync
19791: Procedure WaitMiliSeconds( AMSec : word)
19792: Procedure WaitMS( AMSec : word)');
19793: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
19794: Procedure WideAppend( var dst : WideString; const src : WideString)
19795: Procedure WideAssign( var dst : WideString; var src : WideString)
19796: Procedure WideDelete( var dst : WideString; index, count : Integer)
19797: Procedure WideFree( var s : WideString)
19798: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19799: Procedure WideFromPChar( var dst : WideString; src : PChar)
19800: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19801: Procedure WideSetLength( var dst : WideString; len : Integer)
19802: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
19803: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
```

```
19804: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19805: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19806: Procedure HttpGet(const Url: string; Stream:TStream);
19807: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
19808: Procedure WordWrap1Click( Sender : TObject)
19809: Procedure Write( const AOut : string)
19810: Procedure Write( Socket : TSocket)
19811: procedure Write(S: string);
19812: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19813: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19814: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19815: procedure WriteBuffer(Buffer:String;Count:LongInt)
19816: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19817: Procedure WriteChar( AValue : Char)
19818: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19819: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19820: Procedure WriteFloat( const Section, Name : string; Value : Double)
19821: Procedure WriteHeader( AHeader : TStrings)
19822: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19823: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19824: Procedure WriteLn( const AOut : string)
19825: procedure Writeln(s: string);
19826: Procedure WriteLog( const FileName, LogLine : string)
19827: Procedure WriteRFCReply( AReply : TIdRFCReply)
19828: Procedure WriteRFCStrings( AStrings : TStrings)
19829: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19830: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
19831: Procedure WriteString( const Section, Ident, Value : String)
19832: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19833: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19834: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19835: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19836: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19837: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
19838: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19839: procedure XMLSyntax1Click(Sender: TObject);
19840: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19841: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19842: Procedure ZeroFillStream( Stream : TMemoryStream)
19843: procedure XMLSyntax1Click(Sender: TObject);
19844: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19845: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19846: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19847: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19848: procedure(Sender, Source: TObject;X, Y: Integer)
19849: procedure(Sender, Target: TObject; X, Y: Integer)
19850: procedure(Sender: TObject; ASection, AWidth: Integer)
19851: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
19852: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
19853: procedure(Sender: TObject; var Action: TCloseAction)
19854: procedure(Sender: TObject; var CanClose: Boolean)
19855: procedure(Sender: TObject; var Key: Char);
19856: ProcedureName ProcedureNames ProcedureParametersCursor @
19857:
19858: *************Now Constructors constructor *************
19859:  Size is: 1248 1115 996 628 550 544 501 459 (381)
19860:  Attach( VersionInfoData : Pointer; Size : Integer)
19861:  constructor Create( ABuckets : TBucketListSizes)
19862:  Create( ACallBackWnd : HWND)
19863:  Create( AClient : TCustomTaskDialog)
19864:  Create( AClient : TIdTelnet)
19865:  Create( ACollection : TCollection)
19866:  Create( ACollection : TFavoriteLinkItems)
19867:  Create( ACollection : TTaskDialogButtons)
19868:  Create( AConnection : TIdCustomHTTP)
19869:  Create( ACreateSuspended : Boolean)
19870:  Create( ADataSet : TCustomSQLDataSet)
19871:  CREATE( ADATASET : TDATASET)
19872:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19873:  Create( AGrid : TCustomDBGrid)
19874:  Create( AGrid : TStringGrid; AIndex : Longint)
19875:  Create( AHTTP : TIdCustomHTTP)
19876:  Create( AListItems : TListItems)
19877:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19878:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19879:  Create( AOwner : TCommonCalendar)
19880:  Create( AOwner : TComponent)
19881:  CREATE( AOWNER : TCOMPONENT)
19882:  Create( AOwner : TCustomListView)
19883:  Create( AOwner : TCustomOutline)
19884:  Create( AOwner : TCustomRichEdit)
19885:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19886:  Create( AOwner : TCustomTreeView)
19887:  Create( AOwner : TIdUserManager)
19888:  Create( AOwner : TListItems)
19889:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19890:  CREATE( AOWNER : TPERSISTENT)
19891:  Create( AOwner : TPersistent)
19892:  Create( AOwner : TTable)
```

```
19893:  Create( AOwner : TTreeNodes)
19894:  Create( AOwner : TWinControl; const ClassName : string)
19895:  Create( AParent : TIdCustomHTTP)
19896:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
19897:  Create( AProvider : TBaseProvider)
19898:  Create( AProvider : TCustomProvider);
19899:  Create( AProvider : TDataSetProvider)
19900:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19901:  Create( ASocket : TSocket)
19902:  Create( AStrings : TWideStrings)
19903:  Create( AToolBar : TToolBar)
19904:  Create( ATreeNodes : TTreeNodes)
19905:  Create( Autofill : boolean)
19906:  Create( AWebPageInfo : TAbstractWebPageInfo)
19907:  Create( AWebRequest : TWebRequest)
19908:  Create( Collection : TCollection)
19909:  Create( Collection : TIdMessageParts; ABody : TStrings)
19910:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
19911:  Create( Column : TColumn)
19912:  Create( const AConvFamily : TConvFamily; const ADescription : string)
19913:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
19914:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
19915:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
19916:  Create( const ATabSet : TTabSet)
19917:  Create( const Compensate : Boolean)
19918:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
19919:  Create( const FileName : string)
19920:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
19921:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
19922:  Create( const MaskValue : string)
19923:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19924:  Create( const Prefix : string)
19925:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
19926:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19927:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19928:  Create( CoolBar : TCoolBar)
19929:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
19930:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
19931:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
19932:  Create( DBCtrlGrid : TDBCtrlGrid)
19933:  Create( DSTableProducer : TDSTableProducer)
19934:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
19935:  Create( ErrorCode : DBIResult)
19936:  Create( Field : TBlobField; Mode : TBlobStreamMode)
19937:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
19938:  Create( HeaderControl : TCustomHeaderControl)
19939:  Create( HTTPRequest : TWebRequest)
19940:  Create( iStart : integer; sText : string)
19941:  Create( iValue : Integer)
19942:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19943:  Create( MciErrNo : MCIERROR; const Msg : string)
19944:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19945:  Create( Message : string; ErrorCode : DBResult)
19946:  Create( Msg : string)
19947:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19948:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19949:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19950:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19951:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
19952:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19953:  Create( Owner : TCustomComboBoxEx)
19954:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19955:  Create( Owner : TPersistent)
19956:  Create( Params : TStrings) Create( Size : Cardinal)
19957:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19958:  Create( StatusBar : TCustomStatusBar)
19959:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
19960:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19961:  Create(AHandle:Integer)
19962:  Create(AOwner: TComponent); virtual;
19963:  Create(const AURI : string)
19964:  Create(FileName:String;Mode:Word)
19965:  Create(Instance:THandle;ResName:String;ResType:PChar)
19966:  Create(Stream : TStream)
19967:  Create1( ADataset : TDataset);
19968:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
19969:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19970:  Create2( Other : TObject);
19971:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19972:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19973:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
19974:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19975:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19976:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
```

```
19977:  CreateRes( Ident : Integer);
19978:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
19979:  CreateRes( ResStringRec : PResStringRec);
19980:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
19981:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19982:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19983:  CreateSize( AWidth, AHeight : Integer)
19984:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19985:
19986: ------------------------------------------------------------------------------
19987: unit uPSI_MathMax;
19988: ------------------------------------------------------------------------------
19989:  CONSTS
19990:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
19991:   Cbrt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
19992:   Cbrt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
19993:   Cbrt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
19994:   Cbrt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
19995:   CbrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
19996:   Catalan: Float   = 0.91596559417721901505460351493 24;  // Catalan constant
19997:   PiJ:    Float      = 3.14159265358979323846264338327 95;  // PI
19998:   PI:  Extended =  3.14159265358979323846264338327 95);
19999:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
20000:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
20001:   PiOn4: Float     = 0.78539816339744830961566084581988; // PI / 4
20002:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
20003:   Sqrt3: Float     = 1.73205080756887729352744634150 59;  // Sqrt(3)
20004:   Sqrt5: Float     = 2.23606797749978969640917366873 13;  // Sqrt(5)
20005:   Sqrt10: Float    = 3.16227766016837933199889354443 27;  // Sqrt(10)
20006:   SqrtPi: Float    = 1.77245385090551602729816748334 11;  // Sqrt(PI)
20007:   Sqrt2Pi: Float   = 2.50662827463100050241576528481 1;   // Sqrt(2 * PI)
20008:   TwoPi: Float     = 6.28318530717958647692528676655 9;   // 2 * PI
20009:   ThreePi: Float   = 9.42477796076937971538793014983 85;  // 3 * PI
20010:   Ln2: Float       = 0.69314718055994530941723212145818; // Ln(2)
20011:   Ln10: Float      = 2.30258509299404568401799145468 44;  // Ln(10)
20012:   LnPi: Float      = 1.14472988584940017414342735135 31;  // Ln(PI)
20013:   Log2J: Float     = 0.30102999566398119521373889472449; // Log10(2)
20014:   Log3: Float      = 0.47712125471966243729502790325512; // Log10(3)
20015:   LogPi: Float     = 0.49714987269413385435126828829 09;  // Log10(PI)
20016:   LogE: Float      = 0.43429448190325182765112891891661; // Log10(E)
20017:   E: Float         = 2.71828182845904523536028747135 27;  // Natural constant
20018:   hLn2Pi: Float    = 0.91893853320467274178032973640562; // Ln(2*PI)/2
20019:   inv2Pi: Float    = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
20020:   TwoToPower63: Float = 9223372036854775808.0;          // 2^63
20021:   GoldenMean: Float   = 1.61803398874989484820458683436563 8;  // GoldenMean
20022:   EulerMascheroni: Float = 0.57721566490153286060651209008 24;  // Euler GAMMA
20023:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
20024:   StDelta         : Extended = 0.00001;  {delta for difference equations}
20025:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
20026:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
20027:
20028: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
20029: begin
20030:   MetersPerInch = 0.0254; // [1]
20031:   MetersPerFoot = MetersPerInch * 12;
20032:   MetersPerYard = MetersPerFoot * 3;
20033:   MetersPerMile = MetersPerFoot * 5280;
20034:   MetersPerNauticalMiles = 1852;
20035:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
20036:   MetersPerLightSecond = 2.99792458E8; // [5]
20037:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
20038:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
20039:   MetersPerCubit = 0.4572; // [6][7]
20040:   MetersPerFathom = MetersPerFoot * 6;
20041:   MetersPerFurlong = MetersPerYard * 220;
20042:   MetersPerHand = MetersPerInch * 4;
20043:   MetersPerPace = MetersPerInch * 30;
20044:   MetersPerRod = MetersPerFoot * 16.5;
20045:   MetersPerChain = MetersPerRod * 4;
20046:   MetersPerLink = MetersPerChain / 100;
20047:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
20048:   MetersPerPica = MetersPerPoint * 12;
20049:
20050:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
20051:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
20052:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
20053:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
20054:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
20055:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
20056:
20057:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
20058:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
20059:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
20060:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
20061:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
20062:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
20063:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
20064:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
20065:
```

```
20066:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
20067:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
20068:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
20069:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
20070:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
20071:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
20072:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
20073:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
20074:
20075:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
20076:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
20077:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
20078:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
20079:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
20080:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
20081:
20082:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
20083:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
20084:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
20085:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
20086:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
20087:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
20088:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
20089:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
20090:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
20091:
20092:   GramsPerPound = 453.59237; // [1][7]
20093:   GramsPerDrams = GramsPerPound / 256;
20094:   GramsPerGrains = GramsPerPound / 7000;
20095:   GramsPerTons = GramsPerPound * 2000;
20096:   GramsPerLongTons = GramsPerPound * 2240;
20097:   GramsPerOunces = GramsPerPound / 16;
20098:   GramsPerStones = GramsPerPound * 14;
20099:
20100:   MaxAngle 9223372036854775808.0;
20101:   MaxTanH 5678.26170314707197474596553898 54);
20102:   MaxFactorial( 1754);
20103:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
20104:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
20105:   MaxTanH( 354.89135644669199842162284618659);
20106:   MaxFactorial'LongInt'( 170);
20107:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
20108:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
20109:   MaxTanH( 44.36141955583649980270285577332 3);
20110:   MaxFactorial'LongInt'( 33);
20111:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
20112:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
20113:   PiExt( 3.14159265358979323846264338327 95);
20114:   RatioDegToRad( PiExt / 180.0);
20115:   RatioGradToRad( PiExt / 200.0);
20116:   RatioDegToGrad( 200.0 / 180.0);
20117:   RatioGradToDeg( 180.0 / 200.0);
20118:   Crc16PolynomCCITT'LongWord $1021);
20119:  Crc16PolynomIBM'LongWord $8005);
20120:  Crc16Bits'LongInt'( 16);
20121:  Crc16Bytes'LongInt'( 2);
20122:  Crc16HighBit'LongWord $8000);
20123:  NotCrc16HighBit','LongWord $7FFF);
20124:   Crc32PolynomIEEE','LongWord $04C11DB7);
20125:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
20126:  Crc32Koopman','LongWord $741B8CD7);
20127:  Crc32Bits','LongInt'( 32);
20128:  Crc32Bytes','LongInt'( 4);
20129:  Crc32HighBit','LongWord $80000000);
20130:  NotCrc32HighBit','LongWord $7FFFFFFF);
20131:
20132:   MinByte       = Low(Byte);
20133:   MaxByte       = High(Byte);
20134:   MinWord       = Low(Word);
20135:   MaxWord       = High(Word);
20136:   MinShortInt   = Low(ShortInt);
20137:   MaxShortInt   = High(ShortInt);
20138:   MinSmallInt   = Low(SmallInt);
20139:   MaxSmallInt   = High(SmallInt);
20140:   MinLongWord   = LongWord(Low(LongWord));
20141:   MaxLongWord   = LongWord(High(LongWord));
20142:   MinLongInt    = LongInt(Low(LongInt));
20143:   MaxLongInt    = LongInt(High(LongInt));
20144:   MinInt64      = Int64(Low(Int64));
20145:   MaxInt64      = Int64(High(Int64));
20146:   MinInteger    = Integer(Low(Integer));
20147:   MaxInteger    = Integer(High(Integer));
20148:   MinCardinal   = Cardinal(Low(Cardinal));
20149:   MaxCardinal   = Cardinal(High(Cardinal));
20150:   MinNativeUInt = NativeUInt(Low(NativeUInt));
20151:   MaxNativeUInt = NativeUInt(High(NativeUInt));
20152:   MinNativeInt  = NativeInt(Low(NativeInt));
20153:   MaxNativeInt  = NativeInt(High(NativeInt));
20154:  Function CosH( const Z : Float) : Float;
```

```
20155:  Function SinH( const Z : Float) : Float;
20156:  Function TanH( const Z : Float) : Float;
20157:
20158:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
20159:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
20160:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
20161:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
20162:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
20163:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
20164:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
20165:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
20166:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
20167:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
20168:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
20169:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
20170:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
20171:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
20172:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
20173:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
20174:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
20175:   MaxLog = 709.7827128933840;
20176:   MinLog = -708.3964185322641;
20177:   MaxFac = 170;
20178:   MaxGam = 171.624376956302;
20179:   MaxLgm = 2.556348E+305;
20180:   SingleCompareDelta   = 1.0E-34;
20181:   DoubleCompareDelta   = 1.0E-280;
20182:   {$IFDEF CLR}
20183:   ExtendedCompareDelta = DoubleCompareDelta;
20184:   {$ELSE}
20185:   ExtendedCompareDelta = 1.0E-4400;
20186:   {$ENDIF}
20187:   Bytes1KB  = 1024;
20188:   Bytes1MB  = 1024 * Bytes1KB;
20189:   Bytes1GB  = 1024 * Bytes1MB;
20190:   Bytes64KB = 64 * Bytes1KB;
20191:   Bytes64MB = 64 * Bytes1MB;
20192:   Bytes2GB  = 2 * LongWord(Bytes1GB);
20193:     clBlack32', $FF000000 ));
20194:     clDimGray32', $FF3F3F3F ));
20195:     clGray32', $FF7F7F7F ));
20196:     clLightGray32', $FFBFBFBF ));
20197:     clWhite32', $FFFFFFFF ));
20198:     clMaroon32', $FF7F0000 ));
20199:     clGreen32', $FF007F00 ));
20200:     clOlive32', $FF7F7F00 ));
20201:     clNavy32', $FF00007F ));
20202:     clPurple32', $FF7F007F ));
20203:     clTeal32', $FF007F7F ));
20204:     clRed32', $FFFF0000 ));
20205:     clLime32', $FF00FF00 ));
20206:     clYellow32', $FFFFFF00 ));
20207:     clBlue32', $FF0000FF ));
20208:     clFuchsia32', $FFFF00FF ));
20209:     clAqua32', $FF00FFFF ));
20210:     clAliceBlue32', $FFF0F8FF ));
20211:     clAntiqueWhite32', $FFFAEBD7 ));
20212:     clAquamarine32', $FF7FFFD4 ));
20213:     clAzure32', $FFF0FFFF ));
20214:     clBeige32', $FFF5F5DC ));
20215:     clBisque32', $FFFFE4C4 ));
20216:     clBlancheDalmond32', $FFFFEBCD ));
20217:     clBlueViolet32', $FF8A2BE2 ));
20218:     clBrown32', $FFA52A2A ));
20219:     clBurlyWood32', $FFDEB887 ));
20220:     clCadetblue32', $FF5F9EA0 ));
20221:     clChartReuse32', $FF7FFF00 ));
20222:     clChocolate32', $FFD2691E ));
20223:     clCoral32', $FFFF7F50 ));
20224:     clCornFlowerBlue32', $FF6495ED ));
20225:     clCornSilk32', $FFFFF8DC ));
20226:     clCrimson32', $FFDC143C ));
20227:     clDarkBlue32', $FF00008B ));
20228:     clDarkCyan32', $FF008B8B ));
20229:     clDarkGoldenRod32', $FFB8860B ));
20230:     clDarkGray32', $FFA9A9A9 ));
20231:     clDarkGreen32', $FF006400 ));
20232:     clDarkGrey32', $FFA9A9A9 ));
20233:     clDarkKhaki32', $FFBDB76B ));
20234:     clDarkMagenta32', $FF8B008B ));
20235:     clDarkOliveGreen32', $FF556B2F ));
20236:     clDarkOrange32', $FFFF8C00 ));
20237:     clDarkOrchid32', $FF9932CC ));
20238:     clDarkRed32', $FF8B0000 ));
20239:     clDarkSalmon32', $FFE9967A ));
20240:     clDarkSeaGreen32', $FF8FBC8F ));
20241:     clDarkSlateBlue32', $FF483D8B ));
20242:     clDarkSlateGray32', $FF2F4F4F ));
20243:     clDarkSlateGrey32', $FF2F4F4F ));
```

```
20244:     clDarkTurquoise32', $FF00CED1 ));
20245:     clDarkViolet32', $FF9400D3 ));
20246:     clDeepPink32', $FFFF1493 ));
20247:     clDeepSkyBlue32', $FF00BFFF ));
20248:     clDodgerBlue32', $FF1E90FF ));
20249:     clFireBrick32', $FFB22222 ));
20250:     clFloralWhite32', $FFFFFAF0 ));
20251:     clGainsBoro32', $FFDCDCDC ));
20252:     clGhostWhite32', $FFF8F8FF ));
20253:     clGold32', $FFFFD700 ));
20254:     clGoldenRod32', $FFDAA520 ));
20255:     clGreenYellow32', $FFADFF2F ));
20256:     clGrey32', $FF808080 ));
20257:     clHoneyDew32', $FFF0FFF0 ));
20258:     clHotPink32', $FFFF69B4 ));
20259:     clIndianRed32', $FFCD5C5C ));
20260:     clIndigo32', $FF4B0082 ));
20261:     clIvory32', $FFFFFFF0 ));
20262:     clKhaki32', $FFF0E68C ));
20263:     clLavender32', $FFE6E6FA ));
20264:     clLavenderBlush32', $FFFFF0F5 ));
20265:     clLawnGreen32', $FF7CFC00 ));
20266:     clLemonChiffon32', $FFFFFACD ));
20267:     clLightBlue32', $FFADD8E6 ));
20268:     clLightCoral32', $FFF08080 ));
20269:     clLightCyan32', $FFE0FFFF ));
20270:     clLightGoldenRodYellow32', $FFFAFAD2 ));
20271:     clLightGreen32', $FF90EE90 ));
20272:     clLightGrey32', $FFD3D3D3 ));
20273:     clLightPink32', $FFFFB6C1 ));
20274:     clLightSalmon32', $FFFFA07A ));
20275:     clLightSeagreen32', $FF20B2AA ));
20276:     clLightSkyblue32', $FF87CEFA ));
20277:     clLightSlategray32', $FF778899 ));
20278:     clLightSlategrey32', $FF778899 ));
20279:     clLightSteelblue32', $FFB0C4DE ));
20280:     clLightYellow32', $FFFFFFE0 ));
20281:     clLtGray32', $FFC0C0C0 ));
20282:     clMedGray32', $FFA0A0A4 ));
20283:     clDkGray32', $FF808080 ));
20284:     clMoneyGreen32', $FFC0DCC0 ));
20285:     clLegacySkyBlue32', $FFA6CAF0 ));
20286:     clCream32', $FFFFFBF0 ));
20287:     clLimeGreen32', $FF32CD32 ));
20288:     clLinen32', $FFFAF0E6 ));
20289:     clMediumAquamarine32', $FF66CDAA ));
20290:     clMediumBlue32', $FF0000CD ));
20291:     clMediumOrchid32', $FFBA55D3 ));
20292:     clMediumPurple32', $FF9370DB ));
20293:     clMediumSeaGreen32', $FF3CB371 ));
20294:     clMediumSlateBlue32', $FF7B68EE ));
20295:     clMediumSpringGreen32', $FF00FA9A ));
20296:     clMediumTurquoise32', $FF48D1CC ));
20297:     clMediumVioletRed32', $FFC71585 ));
20298:     clMidnightBlue32', $FF191970 ));
20299:     clMintCream32', $FFF5FFFA ));
20300:     clMistyRose32', $FFFFE4E1 ));
20301:     clMoccasin32', $FFFFE4B5 ));
20302:     clNavajoWhite32', $FFFFDEAD ));
20303:     clOldLace32', $FFFDF5E6 ));
20304:     clOliveDrab32', $FF6B8E23 ));
20305:     clOrange32', $FFFFA500 ));
20306:     clOrangeRed32', $FFFF4500 ));
20307:     clOrchid32', $FFDA70D6 ));
20308:     clPaleGoldenRod32', $FFEEE8AA ));
20309:     clPaleGreen32', $FF98FB98 ));
20310:     clPaleTurquoise32', $FFAFEEEE ));
20311:     clPaleVioletred32', $FFDB7093 ));
20312:     clPapayaWhip32', $FFFFEFD5 ));
20313:     clPeachPuff32', $FFFFDAB9 ));
20314:     clPeru32', $FFCD853F ));
20315:     clPlum32', $FFDDA0DD ));
20316:     clPowderBlue32', $FFB0E0E6 ));
20317:     clRosyBrown32', $FFBC8F8F ));
20318:     clRoyalBlue32', $FF4169E1 ));
20319:     clSaddleBrown32', $FF8B4513 ));
20320:     clSalmon32', $FFFA8072 ));
20321:     clSandyBrown32', $FFF4A460 ));
20322:     clSeaGreen32', $FF2E8B57 ));
20323:     clSeaShell32', $FFFFF5EE ));
20324:     clSienna32', $FFA0522D ));
20325:     clSilver32', $FFC0C0C0 ));
20326:     clSkyblue32', $FF87CEEB ));
20327:     clSlateBlue32', $FF6A5ACD ));
20328:     clSlateGray32', $FF708090 ));
20329:     clSlateGrey32', $FF708090 ));
20330:     clSnow32', $FFFFFAFA ));
20331:     clSpringgreen32', $FF00FF7F ));
20332:     clSteelblue32', $FF4682B4 ));
```

```
20333:     clTan32', $FFD2B48C ));
20334:     clThistle32', $FFD8BFD8 ));
20335:     clTomato32', $FFFF6347 ));
20336:     clTurquoise32', $FF40E0D0 ));
20337:     clViolet32', $FFEE82EE ));
20338:     clWheat32', $FFF5DEB3 ));
20339:     clWhitesmoke32', $FFF5F5F5 ));
20340:     clYellowgreen32', $FF9ACD32 ));
20341:     clTrWhite32', $7FFFFFFF ));
20342:     clTrBlack32', $7F000000 ));
20343:     clTrRed32', $7FFF0000 ));
20344:     clTrGreen32', $7F00FF00 ));
20345:     clTrBlue32', $7F0000FF ));
20346:   // Fixed point math constants
20347:   FixedOne = $10000;  FixedHalf = $7FFF;
20348:   FixedPI  = Round(PI * FixedOne);
20349:   FixedToFloat = 1/FixedOne;
20350:
20351:  Special Types
20352: ***************************************************
20353:   type Complex = record //for complex numbers
20354:     X, Y : Float;
20355:   end;
20356:   type TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R :'
20357:    + Float; Theta : Float; end');
20358:   type TVector     = array of Float;
20359:   TIntVector  = array of Integer;
20360:   TCompVector = array of Complex;
20361:   TBoolVector = array of Boolean;
20362:   TStrVector  = array of String;
20363:   TMatrix     = array of TVector;
20364:   TIntMatrix  = array of TIntVector;
20365:   TCompMatrix = array of TCompVector;
20366:   TBoolMatrix = array of TBoolVector;
20367:   TStrMatrix  = array of TStrVector;
20368:   TByteArray = array[0..32767] of byte; !
20369:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
20370:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
20371:   T2StringArray = array of array of string;
20372:   T2IntegerArray = array of array of integer;
20373:   AddTypeS('INT_PTR', 'Integer
20374:   AddTypeS('LONG_PTR', 'Integer
20375:   AddTypeS('UINT_PTR', 'Cardinal
20376:   AddTypeS('ULONG_PTR', 'Cardinal
20377:   AddTypeS('DWORD_PTR', 'ULONG_PTR
20378:   TIntegerDynArray', 'array of Integer
20379:   TCardinalDynArray', 'array of Cardinal
20380:   TWordDynArray', 'array of Word
20381:   TSmallIntDynArray', 'array of SmallInt
20382:   TByteDynArray', 'array of Byte
20383:   TShortIntDynArray', 'array of ShortInt
20384:   TInt64DynArray', 'array of Int64
20385:   TLongWordDynArray', 'array of LongWord
20386:   TSingleDynArray', 'array of Single
20387:   TDoubleDynArray', 'array of Double
20388:   TBooleanDynArray', 'array of Boolean
20389:   TStringDynArray', 'array of string
20390:   TWideStringDynArray', 'array of WideString
20391:   TDynByteArray     = array of Byte;
20392:   TDynShortintArray = array of Shortint;
20393:   TDynSmallintArray = array of Smallint;
20394:   TDynWordArray     = array of Word;
20395:   TDynIntegerArray  = array of Integer;
20396:   TDynLongintArray  = array of Longint;
20397:   TDynCardinalArray = array of Cardinal;
20398:   TDynInt64Array    = array of Int64;
20399:   TDynExtendedArray = array of Extended;
20400:   TDynDoubleArray   = array of Double;
20401:   TDynSingleArray   = array of Single;
20402:   TDynFloatArray    = array of Float;
20403:   TDynPointerArray  = array of Pointer;
20404:   TDynStringArray   = array of string;
20405:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
20406:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
20407:   TSynSearchOptions = set of TSynSearchOption;
20408:   TFloat = single
20409:   Float = double
20410:
20411:
20412: /* Project: IFSI_WinForm1puzzle.pas BaseInclude RunTimeLib for maXbox *: pas_includebox.inc
20413: ---------------------------------------------------------------------------
20414: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
20415: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
20416: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
20417: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
20418: function CheckStringSum(vstring: string): integer;
20419: function HexToInt(HexNum: string): LongInt;
20420: function IntToBin(Int: Integer): String;
20421: function BinToInt(Binary: String): Integer;
```

```
20422: function HexToBin(HexNum: string): string; external2
20423: function BinToHex(Binary: String): string;
20424: function IntToFloat(i: Integer): double;
20425: function AddThousandSeparator(S: string; myChr: Char): string;
20426: function Max3(const X,Y,Z: Integer): Integer;
20427: procedure Swap(var X,Y: char); // faster without inline
20428: procedure ReverseString(var S: String);
20429: function CharToHexStr(Value: Char): string;
20430: function CharToUniCode(Value: Char): string;
20431: function Hex2Dec(Value: Str002): Byte;
20432: function HexStrCodeToStr(Value: string): string;
20433: function HexToStr(i: integer; value: string): string;
20434: function UniCodeToStr(Value: string): string;
20435: function CRC16(statement: string): string;
20436: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
20437: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
20438: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
20439: Procedure ExecuteCommand(executeFile, paramstring: string);
20440: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
20441: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
20442: procedure SearchAndOpenDoc(vfilenamepath: string);
20443: procedure ShowInterfaces(myFile: string);
20444: function Fact2(av: integer): extended;
20445: Function BoolToStr(B: Boolean): string;
20446: Function GCD(x, y : LongInt) : LongInt;
20447: function LCM(m,n: longint): longint;
20448: function GetASCII: string;
20449: function GetItemHeight(Font: TFont): Integer;
20450: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
20451: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
20452: function getHINSTANCE: longword;
20453: function getHMODULE: longword;
20454: function GetASCII: string;
20455: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
20456: function WordIsOk(const AWord: string; var VW: Word): boolean;
20457: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
20458: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
20459: function SafeStr(const s: string): string;
20460: function ExtractUrlPath(const FileName: string): string;
20461: function ExtractUrlName(const FileName: string): string;
20462: function IsInternet: boolean;
20463: function RotateLeft1Bit_u32( Value: uint32): uint32;
20464: procedure LinearRegression(const KnownY:array of Double;const KnownX:array of Double;NData:Int;var
      LF:TStLinEst; ErrorStats: Bool);
20465: procedure getEnvironmentInfo;
20466: procedure AntiFreeze;
20467: function GetCPUSpeed: Double;
20468: function IsVirtualPcGuest : Boolean;
20469: function IsVmWareGuest : Boolean;
20470: procedure StartSerialDialog;
20471: function IsWoW64: boolean;
20472: function IsWow64String(var s: string): Boolean;
20473: procedure StartThreadDemo;
20474: Function RGB(R,G,B: Byte): TColor;
20475: Function Sendln(amess: string): boolean;
20476: Procedure maXbox;
20477: Function AspectRatio(aWidth, aHeight: Integer): String;
20478: function wget(aURL, afile: string): boolean;
20479: procedure PrintList(Value: TStringList);
20480: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
20481: procedure getEnvironmentInfo;
20482: procedure AntiFreeze;
20483: function getBitmap(apath: string): TBitmap;
20484: procedure ShowMessageBig(const aText : string);
20485: function YesNoDialog(const ACaption, AMsg: string): boolean;
20486: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
20487: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
20488: //function myStrToBytes(const Value: String): TBytes;
20489: //function myBytesToStr(const Value: TBytes): String;
20490: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20491: function getBitmap(apath: string): TBitmap;
20492: procedure ShowMessageBig(const aText : string);
20493: Function StrToBytes(const Value: String): TBytes;
20494: Function BytesToStr(const Value: TBytes): String;
20495: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20496: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
      HostName:String):Bool;
20497: function FindInPaths(const fileName, paths : String) : String;
20498: procedure initHexArray(var hexn: THexArray);
20499: function josephusG(n,k: integer; var graphout: string): integer;
20500: function isPowerof2(num: int64): boolean;
20501: function powerOf2(exponent: integer): int64;
20502: function getBigPI: string;
20503: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
20504:  function GetASCIILine: string;
20505: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
20506:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
20507: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
20508: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
```

```
20509: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
20510: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
20511: function isKeypressed: boolean;
20512: function Keypress: boolean;
20513: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
20514: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
20515: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
20516: function GetOSName: string;
20517: function GetOSVersion: string;
20518: function GetOSNumber: string;
20519: function getEnvironmentString: string;
20520: procedure StrReplace(var Str: String; Old, New: String);
20521: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
20522: function getTeamViewerID: string;
20523: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
20524: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
20525: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
20526: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
20527: function StartSocketService: Boolean;
20528: procedure StartSocketServiceForm;
20529: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
20530: function GetFileList1(apath: string): TStringlist;
20531: procedure LetFileList(FileList: TStringlist; apath: string);
20532: procedure StartWeb(aurl: string);
20533: function GetTodayFiles(startdir, amask: string): TStringlist;
20534: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
20535: function JavahashCode(val: string): Integer;
20536: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
20537: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
20538: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
20539: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
20540: Procedure ConvertToGray(Cnv: TCanvas);
20541: function GetFileDate(aFile:string; aWithTime:Boolean):string;
20542: procedure ShowMemory;
20543: function ShowMemory2: string;
20544: function getHostIP: string;
20545: procedure ShowBitmap(bmap: TBitmap);
20546: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
20547: function CreateDBGridForm(dblist: TStringList): TListbox;
20548: function isService: boolean;
20549: function isApplication: boolean;
20550: function isTerminalSession: boolean;
20551:  function SetPrivilege(privilegeName: string; enable: boolean): boolean;
20552:   procedure getScriptandRunAsk;
20553:   procedure getScriptandRun(ascript: string);
20554:   function VersionCheckAct: string;
20555:   procedure getBox(aurl, extension: string);
20556:   function CheckBox: string;
20557:   function isNTFS: boolean;
20558:   //procedure doWebCamPic;
20559:   procedure doWebCamPic(picname: string);
20560:   function readm: string;
20561:   procedure getGEOMapandRunAsk;
20562:  function GetMapX(C_form,apath: string; const Data: string): boolean;
20563:   procedure GetGEOMap(C_form,apath: string; const Data: string);
20564:   function GetMapXGeoReverse(C_form: string; const lat,long: string): string;
20565:   //function RoundTo(const AValue: Extended;
20566:     //             const ADigit: TRoundToEXRangeExtended): Extended;
20567:  function DownloadFile(SourceFile, DestFile: string): Boolean;
20568:  function DownloadFileOpen(SourceFile, DestFile: string): Boolean;
20569:  function OpenMap(const Data: string): boolean;
20570:  function GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
20571:  Function getFileCount(amask: string): integer;
20572:  function CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TNavPos): string;
20573:  procedure Debugln(DebugLOGFILE: string; E: string);
20574:  function IntToFloat(i: Integer): double;
20575:  function AddThousandSeparator(S: string; myChr: Char): string;
20576:  function mymciSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
20577:  End C:\maXbook\maxbox3\mX3999\maxbox3\source\IFSI_WinForm1puzzle.pas File loaded
20578:
20579:
20580: // News of 3.9.8 up
20581: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20582: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20583: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20584: JvChart - TJvChart Component - 2009 Public
20585: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
20586: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
20587: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
20588: DMath DLL included incl. Demos
20589: Interface Navigator menu/View/Intf Navigator
20590: Unit Explorer menu/Debug/Units Explorer
20591: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
20592: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
20593: Script History to 9 Files WebServer light /Options/Addons/WebServer
20594: Full Text Finder, JVSimLogic Simulator Package
20595: Halt-Stop Program in Menu, WebServer2, Stop Event ,
20596: Conversion Routines, Prebuild Forms, CodeSearch
20597: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
```

```
20598: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20599: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20600: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
20601: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
20602: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
20603: IDE Reflection API, Session Service Shell S3
20604: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
20605: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
20606: arduino map() function, PMRandom Generator
20607: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
20608: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
20609: REST Test Lib, Multilang Component, Forth Interpreter
20610: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
20611: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
20612: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20613: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20614: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
20615: QRCode Service, add more CFunctions like CDateTime of Synapse
20616: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
20617: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
20618: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
20619: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
20620: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
20621: BOLD Package, Indy Package5, maTRIx. MATHEMAX
20622: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
20623: emax layers: system-package-component-unit-class-function-block
20624: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
20625: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
20626: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
20627: OpenGL Game Demo: ..Options/Add Ons/Reversi
20628: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
20629: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
20630: 7% performance gain (hot spot profiling)
20631: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
20632: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
20633: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
20634: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
20635:
20636: add routines in 3.9.7.5
20637: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
20638: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
20639: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
20640: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
20641: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
20642: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
20643: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
20644:
20645: ////////////////////////// TestUnits //////////////////////////
20646:   SelftestPEM;
20647:   SelfTestCFundamentUtils;
20648:   SelfTestCFileUtils;
20649:   SelfTestCDateTime;
20650:   SelfTestCTimer;
20651:   SelfTestCRandom;
20652:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
20653:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
20654:
20655:   // Note: There's no need for installing a client certificate in the
20656:   //       webbrowser. The server asks the webbrowser to send a certificate but
20657:   //       if nothing is installed the software will work because the server
20658:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
20659:   //       file: c_cacert.p12  password: c_cakey
20660:
20661:   TGraphicControl = class(TControl)
20662:   private
20663:     FCanvas: TCanvas;
20664:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20665:   protected
20666:     procedure Paint; virtual;
20667:     property Canvas: TCanvas read FCanvas;
20668:   public
20669:     constructor Create(AOwner: TComponent); override;
20670:     destructor Destroy; override;
20671:   end;
20672:
20673:   TCustomControl = class(TWinControl)
20674:   private
20675:     FCanvas: TCanvas;
20676:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20677:   protected
20678:     procedure Paint; virtual;
20679:     procedure PaintWindow(DC: HDC); override;
20680:     property Canvas: TCanvas read FCanvas;
20681:   public
20682:     constructor Create(AOwner: TComponent); override;
20683:     destructor Destroy; override;
20684:   end;
20685:     RegisterPublishedProperties;
20686:     ('ONCHANGE', 'TNotifyEvent', iptrw);
```

```
20687:     ('ONCLICK', 'TNotifyEvent', iptrw);
20688:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
20689:     ('ONENTER', 'TNotifyEvent', iptrw);
20690:     ('ONEXIT', 'TNotifyEvent', iptrw);
20691:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
20692:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
20693:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
20694:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
20695:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
20696: //*********************************************************************
20697:   // To stop the while loop, click on Options/Show Include (boolean switch)!
20698:   Control a loop in a script with a form event:
20699:   IncludeON;   //control the while loop
20700:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
20701:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
20702:
20703: //---------------------------------------------------------------------------
20704: //*************mX4 ini-file Configuration***********************************
20705: //---------------------------------------------------------------------------
20706:   using config file maxboxdef.ini       menu/Help/Config File
20707:
20708: //*** Definitions for maXbox mX3 ***
20709: [FORM]
20710: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
20711: FONTSIZE=14
20712: EXTENSION=txt
20713: SCREENX=1386
20714: SCREENY=1077
20715: MEMHEIGHT=350
20716: PRINTFONT=Courier New //GUI Settings
20717: LINENUMBERS=Y   //line numbers at gutter in editor at left side
20718: EXCEPTIONLOG=Y  //store excepts + success in 2 log files see below! –menu Debug/Show Last Exceptions
20719: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
20720: BOOTSCRIPT=Y    //enabling load a boot script
20721: MEMORYREPORT=Y  //shows memory report on closing maXbox
20722: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
20723: NAVIGATOR=N     //shows function list at the right side of editor
20724: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
20725: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
20726: [WEB]
20727: IPPORT=8080     //for internal webserver – menu /Options/Add Ons/WebServer
20728: IPHOST=192.168.1.53   //run as Administrator!
20729: ROOTCERT=filepathY
20730: SCERT=filepathY
20731: RSAKEY=filepathY
20732: VERSIONCHECK=Y
20733: APP=C:\WINDOWS\System32\calc.exe  //set path to an external app
20734:
20735:
20736: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
20737: Also possible to set report memory in script to override ini setting
20738: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
20739:
20740: After Change the ini file you can reload the file with ../Help/Config Update
20741:
20742: //---------------------------------------------------------------------------
20743: //*************mX4 maildef.ini ini-file Configuration*********************
20744: //---------------------------------------------------------------------------
20745: //*** Definitions for maXMail ***
20746: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
20747: [MAXMAIL]
20748: HOST=getmail.softwareschule.ch
20749: USER=mailusername
20750: PASS=password
20751: PORT=110
20752: SSL=Y
20753: BODY=Y
20754: LAST=5
20755:
20756: ADO Connection String:
20757: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
20758: \452_dbtreeview2access.txt \452_dbtrv3accessUML2.txt
20759: program TestDbTreeViewMainForm2_ACCESS;
20760:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20761:                     +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20762:   'Provider=MSDASQL.1;Persist Security Info=False;Extended
      Properties="DSN=FB_EMPLOYEE;Driver=Firebird/InterBase(r)
      driver;Dbname=C:\maXbook\maxbox3\mX3999\maxbox3\examples\EMPLOYEE.FDB;CHARSET=NONE;UID=SYSDBA;Role=Admin;"'
20763:
20764: OpenSSL Lib:  unit ssl_openssl_lib;
20765: {$IFDEF CIL}
20766: const
20767:   {$IFDEF LINUX}
20768:   DLLSSLName = 'libssl.so';
20769:   DLLUtilName = 'libcrypto.so';
20770:   {$ELSE}
20771:   DLLSSLName = 'ssleay32.dll';
20772:   DLLUtilName = 'libeay32.dll';
20773:   {$ENDIF}
```

```
20774: {$ELSE}
20775: var
20776:   {$IFNDEF MSWINDOWS}
20777:     {$IFDEF DARWIN}
20778:     DLLSSLName: string = 'libssl.dylib';
20779:     DLLUtilName: string = 'libcrypto.dylib';
20780:     {$ELSE}
20781:     DLLSSLName: string = 'libssl.so';
20782:     DLLUtilName: string = 'libcrypto.so';
20783:     {$ENDIF}
20784:   {$ELSE}
20785:   DLLSSLName: string = 'ssleay32.dll';
20786:   DLLSSLName2: string = 'libssl32.dll';
20787:   DLLUtilName: string = 'libeay32.dll';
20788:   {$ENDIF}
20789: {$ENDIF}
20790:
20791: //-------------------------------------------------------------------------
20792: //*************mX4 Macro Tags  ******************************************
20793: //-------------------------------------------------------- ----------------
20794:
20795:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
      E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
20796:
20797: //Tag Macros in ini-file configure
20798:
20799:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20800:
20801: //Tag Macros
20802: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20803: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20804: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
20805: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20806: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20807: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
20808: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20809: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
20810: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
20811:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
20812: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
20813: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20814:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20815: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
20816:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
20817: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
20818:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
20819:
20820: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20821:
20822: //Replace Macros
20823:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20824:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20825:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20826:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
20827:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
20828:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
20829:   ref: netcologne.dl.sourceforge.net/project/maxbox/maxbox3.zip
20830:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20831:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
20832: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20833:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
20834:
20835: //-------------------------------------------------------------------------
20836: //*************mX4 ToDo List Tags  ../Help/ToDo List**********************
20837: //-------------------------------------------------------- ----------------
20838:
20839:     while I < sl.Count do begin
20840: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
20841:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
20842:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
20843:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
20844:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20845:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
20846:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20847:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
20848:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20849:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
20850:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
20851:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
20852:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
20853:       Inc(I);
20854:     end;
20855:
20856: //-------------------------------------------------------------------------
20857: //*************mX4 Public  Tools API ************************************
20858: //-------------------------------------------------------------------------
20859:   file : unit uPSI_fMain.pas;               {$OTAP} Open Tools API Catalog
20860:  // Those functions concern the editor and preprocessor, all of the IDE
20861:  Example: Call it with maxform1.Info1Click(self)
```

```
20862:  Note: Call all Methods with maxForm1., e.g.:
20863:                        maxForm1.ShellStyle1Click(self);
20864:
20865: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20866: begin
20867:  Const('BYTECODE','String 'bytecode.txt'
20868:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
20869:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
20870:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
20871:  Const('PSINC','String PS Includes (*.inc)|*.INC
20872:  Const('DEFFILENAME','String 'firstdemo.txt
20873:  Const('DEFINIFILE','String 'maxboxdef.ini
20874:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
20875:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
20876:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
20877:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
20878:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
20879:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
20880:  Const('INCLUDEBOX','String 'pas_includebox.inc
20881:  Const('BOOTSCRIPT','String 'maxbootscript.txt
20882:  Const('MBVERSION','String '3.9.9.160
20883:  Const('VERSION','String'3.9.9.160
20884:  Const('MBVER','String '399
20885:  Const('MBVERI','Integer'(399);
20886:  Const('MBVERIALL','Integer'(399160);
20887:  Const('EXENAME','String 'maXbox3.exe
20888:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
20889:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
20890:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
20891:  Const('MXINTERNETCHECK','String 'www.ask.com
20892:  Const('MXMAIL','String 'max@kleiner.com
20893:  Const('TAB','Char #$09);
20894:  Const('CODECOMPLETION','String 'bds_delphi.dci
20895:  SIRegister_TMaxForm1(CL);
20896: end;
20897:
20898:   with FindClass('TForm'),'TMaxForm1') do begin
20899:     memo2', 'TMemo', iptrw);
20900:     memo1', 'TSynMemo', iptrw);
20901:     CB1SCList', 'TComboBox', iptrw);
20902:     mxNavigator', 'TComboBox', iptrw);
20903:     IPHost', 'string', iptrw);
20904:     IPPort', 'integer', iptrw);
20905:     COMPort', 'integer', iptrw);      //3.9.6.4
20906:     Splitter1', 'TSplitter', iptrw);
20907:     PSScript', 'TPSScript', iptrw);
20908:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
20909:     MainMenu1', 'TMainMenu', iptrw);
20910:     Program1', 'TMenuItem', iptrw);
20911:     Compile1', 'TMenuItem', iptrw);
20912:     Files1', 'TMenuItem', iptrw);
20913:     open1', 'TMenuItem', iptrw);
20914:     Save2', 'TMenuItem', iptrw);
20915:     Options1', 'TMenuItem', iptrw);
20916:     Savebefore1', 'TMenuItem', iptrw);
20917:     Largefont1', 'TMenuItem', iptrw);
20918:     sBytecode1', 'TMenuItem', iptrw);
20919:     Saveas3', 'TMenuItem', iptrw);
20920:     Clear1', 'TMenuItem', iptrw);
20921:     Slinenumbers1', 'TMenuItem', iptrw);
20922:     About1', 'TMenuItem', iptrw);
20923:     Search1', 'TMenuItem', iptrw);
20924:     SynPasSyn1', 'TSynPasSyn', iptrw);
20925:     memo1', 'TSynMemo', iptrw);
20926:     SynEditSearch1', 'TSynEditSearch', iptrw);
20927:     WordWrap1', 'TMenuItem', iptrw);
20928:     XPManifest1', 'TXPManifest', iptrw);
20929:     SearchNext1', 'TMenuItem', iptrw);
20930:     Replace1', 'TMenuItem', iptrw);
20931:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
20932:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
20933:     ShowInclude1', 'TMenuItem', iptrw);
20934:     SynEditPrint1', 'TSynEditPrint', iptrw);
20935:     Printout1', 'TMenuItem', iptrw);
20936:     mnPrintColors1', 'TMenuItem', iptrw);
20937:     dlgFilePrint', 'TPrintDialog', iptrw);
20938:     dlgPrintFont1', 'TFontDialog', iptrw);
20939:     mnuPrintFont1', 'TMenuItem', iptrw);
20940:     Include1', 'TMenuItem', iptrw);
20941:     CodeCompletionList1', 'TMenuItem', iptrw);
20942:     IncludeList1', 'TMenuItem', iptrw);
20943:     ImageList1', 'TImageList', iptrw);
20944:     ImageList2', 'TImageList', iptrw);
20945:     CoolBar1', 'TCoolBar', iptrw);
20946:     ToolBar1', 'TToolBar', iptrw);
20947:     tbtnLoad', 'TToolButton', iptrw);
20948:     ToolButton2', 'TToolButton', iptrw);
20949:     tbtnFind', 'TToolButton', iptrw);
20950:     tbtnCompile', 'TToolButton', iptrw);
```

```
20951:     tbtnTrans', 'TToolButton', iptrw);
20952:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
20953:     toolbtnTutorial', 'TToolButton', iptrw);
20954:     tbtn6res', 'TToolButton', iptrw);
20955:     ToolButton5', 'TToolButton', iptrw);
20956:     ToolButton1', 'TToolButton', iptrw);
20957:     ToolButton3', 'TToolButton', iptrw);
20958:     statusBar1', 'TStatusBar', iptrw);
20959:     SaveOutput1', 'TMenuItem', iptrw);
20960:     ExportClipboard1', 'TMenuItem', iptrw);
20961:     Close1', 'TMenuItem', iptrw);
20962:     Manual1', 'TMenuItem', iptrw);
20963:     About2', 'TMenuItem', iptrw);
20964:     loadLastfile1', 'TMenuItem', iptrw);
20965:     imglogo', 'TImage', iptrw);
20966:     cedebug', 'TPSScriptDebugger', iptrw);
20967:     debugPopupMenu1', 'TPopupMenu', iptrw);
20968:     BreakPointMenu', 'TMenuItem', iptrw);
20969:     Decompile1', 'TMenuItem', iptrw);
20970:     StepInto1', 'TMenuItem', iptrw);
20971:     StepOut1', 'TMenuItem', iptrw);
20972:     Reset1', 'TMenuItem', iptrw);
20973:     DebugRun1', 'TMenuItem', iptrw);
20974:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20975:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20976:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
20977:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20978:     tutorial4', 'TMenuItem', iptrw);
20979:     ExporttoClipboard1', 'TMenuItem', iptrw);
20980:     ImportfromClipboard1', 'TMenuItem', iptrw);
20981:     N4', 'TMenuItem', iptrw);
20982:     N5', 'TMenuItem', iptrw);
20983:     N6', 'TMenuItem', iptrw);
20984:     ImportfromClipboard2', 'TMenuItem', iptrw);
20985:     tutorial1', 'TMenuItem', iptrw);
20986:     N7', 'TMenuItem', iptrw);
20987:     ShowSpecChars1', 'TMenuItem', iptrw);
20988:     OpenDirectory1', 'TMenuItem', iptrw);
20989:     procMess', 'TMenuItem', iptrw);
20990:     tbtnUseCase', 'TToolButton', iptrw);
20991:     ToolButton7', 'TToolButton', iptrw);
20992:     EditFont1', 'TMenuItem', iptrw);
20993:     UseCase1', 'TMenuItem', iptrw);
20994:     tutorial21', 'TMenuItem', iptrw);
20995:     OpenUseCase1', 'TMenuItem', iptrw);
20996:     PSImport_DB1', 'TPSImport_DB', iptrw);
20997:     tutorial31', 'TMenuItem', iptrw);
20998:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20999:     HTMLSyntax1', 'TMenuItem', iptrw);
21000:     ShowInterfaces1', 'TMenuItem', iptrw);
21001:     Tutorial5', 'TMenuItem', iptrw);
21002:     AllFunctionsList1', 'TMenuItem', iptrw);
21003:     ShowLastException1', 'TMenuItem', iptrw);
21004:     PlayMP31', 'TMenuItem', iptrw);
21005:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
21006:     texSyntax1', 'TMenuItem', iptrw);
21007:     N8', 'TMenuItem', iptrw);
21008:     GetEMails1', 'TMenuItem', iptrw);
21009:     SynCppSyn1', 'TSynCppSyn', iptrw);
21010:     CSyntax1', 'TMenuItem', iptrw);
21011:     Tutorial6', 'TMenuItem', iptrw);
21012:     New1', 'TMenuItem', iptrw);
21013:     AllObjectsList1', 'TMenuItem', iptrw);
21014:     LoadBytecode1', 'TMenuItem', iptrw);
21015:     CipherFile1', 'TMenuItem', iptrw);
21016:     N9', 'TMenuItem', iptrw);
21017:     N10', 'TMenuItem', iptrw);
21018:     Tutorial11', 'TMenuItem', iptrw);
21019:     Tutorial71', 'TMenuItem', iptrw);
21020:     UpdateService1', 'TMenuItem', iptrw);
21021:     PascalSchool1', 'TMenuItem', iptrw);
21022:     Tutorial81', 'TMenuItem', iptrw);
21023:     DelphiSite1', 'TMenuItem', iptrw);
21024:     Output1', 'TMenuItem', iptrw);
21025:     TerminalStyle1', 'TMenuItem', iptrw);
21026:     ReadOnly1', 'TMenuItem', iptrw);
21027:     ShellStyle1', 'TMenuItem', iptrw);
21028:     BigScreen1', 'TMenuItem', iptrw);
21029:     Tutorial91', 'TMenuItem', iptrw);
21030:     SaveOutput2', 'TMenuItem', iptrw);
21031:     N11', 'TMenuItem', iptrw);
21032:     SaveScreenshot', 'TMenuItem', iptrw);
21033:     Tutorial101', 'TMenuItem', iptrw);
21034:     SQLSyntax1', 'TMenuItem', iptrw);
21035:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
21036:     Console1', 'TMenuItem', iptrw);
21037:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
21038:     XMLSyntax1', 'TMenuItem', iptrw);
21039:     ComponentCount1', 'TMenuItem', iptrw);
```

```
21040:     NewInstance1', 'TMenuItem', iptrw);
21041:     toolbtnTutorial', 'TToolButton', iptrw);
21042:     Memory1', 'TMenuItem', iptrw);
21043:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
21044:     JavaSyntax1', 'TMenuItem', iptrw);
21045:     SyntaxCheck1', 'TMenuItem', iptrw);
21046:     Tutorial10Statistics1', 'TMenuItem', iptrw);
21047:     ScriptExplorer1', 'TMenuItem', iptrw);
21048:     FormOutput1', 'TMenuItem', iptrw);
21049:     ArduinoDump1', 'TMenuItem', iptrw);
21050:     AndroidDump1', 'TMenuItem', iptrw);
21051:     GotoEnd1', 'TMenuItem', iptrw);
21052:     AllResourceList1', 'TMenuItem', iptrw);
21053:     ToolButton4', 'TToolButton', iptrw);
21054:     tbtn6res', 'TToolButton', iptrw);
21055:     Tutorial11Forms1', 'TMenuItem', iptrw);
21056:     Tutorial12SQL1', 'TMenuItem', iptrw);
21057:     ResourceExplore1', 'TMenuItem', iptrw);
21058:     Info1', 'TMenuItem', iptrw);
21059:     N12', 'TMenuItem', iptrw);
21060:     CryptoBox1', 'TMenuItem', iptrw);
21061:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
21062:     CipherFile2', 'TMenuItem', iptrw);
21063:     N13', 'TMenuItem', iptrw);
21064:     ModulesCount1', 'TMenuItem', iptrw);
21065:     AddOns2', 'TMenuItem', iptrw);
21066:     N4GewinntGame1', 'TMenuItem', iptrw);
21067:     DocuforAddOns1', 'TMenuItem', iptrw);
21068:     Tutorial14Async1', 'TMenuItem', iptrw);
21069:     Lessons15Review1', 'TMenuItem', iptrw);
21070:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
21071:     PHPSyntax1', 'TMenuItem', iptrw);
21072:     Breakpoint1', 'TMenuItem', iptrw);
21073:     SerialRS2321', 'TMenuItem', iptrw);
21074:     N14', 'TMenuItem', iptrw);
21075:     SynCSSyn1', 'TSynCSSyn', iptrw);
21076:     CSyntax2', 'TMenuItem', iptrw);
21077:     Calculator1', 'TMenuItem', iptrw);
21078:     tbtnSerial', 'TToolButton', iptrw);
21079:     ToolButton8', 'TToolButton', iptrw);
21080:     Tutorial151', 'TMenuItem', iptrw);
21081:     N15', 'TMenuItem', iptrw);
21082:     N16', 'TMenuItem', iptrw);
21083:     ControlBar1', 'TControlBar', iptrw);
21084:     ToolBar2', 'TToolBar', iptrw);
21085:     BtnOpen', 'TToolButton', iptrw);
21086:     BtnSave', 'TToolButton', iptrw);
21087:     BtnPrint', 'TToolButton', iptrw);
21088:     BtnColors', 'TToolButton', iptrw);
21089:     btnClassReport', 'TToolButton', iptrw);
21090:     BtnRotateRight', 'TToolButton', iptrw);
21091:     BtnFullSize', 'TToolButton', iptrw);
21092:     BtnFitToWindowSize', 'TToolButton', iptrw);
21093:     BtnZoomMinus', 'TToolButton', iptrw);
21094:     BtnZoomPlus', 'TToolButton', iptrw);
21095:     Panel1', 'TPanel', iptrw);
21096:     LabelBrettgroesse', 'TLabel', iptrw);
21097:     CB1SCList', 'TComboBox', iptrw);
21098:     ImageListNormal', 'TImageList', iptrw);
21099:     spbtnexplore', 'TSpeedButton', iptrw);
21100:     spbtnexample', 'TSpeedButton', iptrw);
21101:     spbsaveas', 'TSpeedButton', iptrw);
21102:     imglogobox', 'TImage', iptrw);
21103:     EnlargeFont1', 'TMenuItem', iptrw);
21104:     EnlargeFont2', 'TMenuItem', iptrw);
21105:     ShrinkFont1', 'TMenuItem', iptrw);
21106:     ThreadDemo1', 'TMenuItem', iptrw);
21107:     HEXEditor1', 'TMenuItem', iptrw);
21108:     HEXView1', 'TMenuItem', iptrw);
21109:     HEXInspect1', 'TMenuItem', iptrw);
21110:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
21111:     ExporttoHTML1', 'TMenuItem', iptrw);
21112:     ClassCount1', 'TMenuItem', iptrw);
21113:     HTMLOutput1', 'TMenuItem', iptrw);
21114:     HEXEditor2', 'TMenuItem', iptrw);
21115:     Minesweeper1', 'TMenuItem', iptrw);
21116:     N17', 'TMenuItem', iptrw);
21117:     PicturePuzzle1', 'TMenuItem', iptrw);
21118:     sbvclhelp', 'TSpeedButton', iptrw);
21119:     DependencyWalker1', 'TMenuItem', iptrw);
21120:     WebScanner1', 'TMenuItem', iptrw);
21121:     View1', 'TMenuItem', iptrw);
21122:     mnToolbar1', 'TMenuItem', iptrw);
21123:     mnStatusbar2', 'TMenuItem', iptrw);
21124:     mnConsole2', 'TMenuItem', iptrw);
21125:     mnCoolbar2', 'TMenuItem', iptrw);
21126:     mnSplitter2', 'TMenuItem', iptrw);
21127:     WebServer1', 'TMenuItem', iptrw);
21128:     Tutorial17Server1', 'TMenuItem', iptrw);
```

```
21129:     Tutorial18Arduino1', 'TMenuItem', iptrw);
21130:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
21131:     PerlSyntax1', 'TMenuItem', iptrw);
21132:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
21133:     PythonSyntax1', 'TMenuItem', iptrw);
21134:     DMathLibrary1', 'TMenuItem', iptrw);
21135:     IntfNavigator1', 'TMenuItem', iptrw);
21136:     EnlargeFontConsole1', 'TMenuItem', iptrw);
21137:     ShrinkFontConsole1', 'TMenuItem', iptrw);
21138:     SetInterfaceList1', 'TMenuItem', iptrw);
21139:     popintfList', 'TPopupMenu', iptrw);
21140:     intfAdd1', 'TMenuItem', iptrw);
21141:     intfDelete1', 'TMenuItem', iptrw);
21142:     intfRefactor1', 'TMenuItem', iptrw);
21143:     Defactor1', 'TMenuItem', iptrw);
21144:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
21145:     Tutorial20Regex', 'TMenuItem', iptrw);
21146:     N18', 'TMenuItem', iptrw);
21147:     ManualE1', 'TMenuItem', iptrw);
21148:     FullTextFinder1', 'TMenuItem', iptrw);
21149:     Move1', 'TMenuItem', iptrw);
21150:     FractalDemo1', 'TMenuItem', iptrw);
21151:     Tutorial21Android1', 'TMenuItem', iptrw);
21152:     Tutorial0Function1', 'TMenuItem', iptrw);
21153:     SimuLogBox1', 'TMenuItem', iptrw);
21154:     OpenExamples1', 'TMenuItem', iptrw);
21155:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
21156:     JavaScriptSyntax1', 'TMenuItem', iptrw);
21157:     Halt1', 'TMenuItem', iptrw);
21158:     CodeSearch1', 'TMenuItem', iptrw);
21159:     SynRubySyn1', 'TSynRubySyn', iptrw);
21160:     RubySyntax1', 'TMenuItem', iptrw);
21161:     Undo1', 'TMenuItem', iptrw);
21162:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
21163:     LinuxShellScript1', 'TMenuItem', iptrw);
21164:     Rename1', 'TMenuItem', iptrw);
21165:     spdcodesearch', 'TSpeedButton', iptrw);
21166:     Preview1', 'TMenuItem', iptrw);
21167:     Tutorial22Services1', 'TMenuItem', iptrw);
21168:     Tutorial23RealTime1', 'TMenuItem', iptrw);
21169:     Configuration1', 'TMenuItem', iptrw);
21170:     MP3Player1', 'TMenuItem', iptrw);
21171:     DLLSpy1', 'TMenuItem', iptrw);
21172:     SynURIOpener1', 'TSynURIOpener', iptrw);
21173:     SynURISyn1', 'TSynURISyn', iptrw);
21174:     URILinksClicks1', 'TMenuItem', iptrw);
21175:     EditReplace1', 'TMenuItem', iptrw);
21176:     GotoLine1', 'TMenuItem', iptrw);
21177:     ActiveLineColor1', 'TMenuItem', iptrw);
21178:     ConfigFile1', 'TMenuItem', iptrw);
21179:     Sort1Intflist', 'TMenuItem', iptrw);
21180:     Redo1', 'TMenuItem', iptrw);
21181:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
21182:     Tutorial25Configuration1', 'TMenuItem', iptrw);
21183:     IndentSelection1', 'TMenuItem', iptrw);
21184:     UnindentSection1', 'TMenuItem', iptrw);
21185:     SkyStyle1', 'TMenuItem', iptrw);
21186:     N19', 'TMenuItem', iptrw);
21187:     CountWords1', 'TMenuItem', iptrw);
21188:     imbookmarkimages', 'TImageList', iptrw);
21189:     Bookmark11', 'TMenuItem', iptrw);
21190:     N20', 'TMenuItem', iptrw);
21191:     Bookmark21', 'TMenuItem', iptrw);
21192:     Bookmark31', 'TMenuItem', iptrw);
21193:     Bookmark41', 'TMenuItem', iptrw);
21194:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
21195:
21196:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
21197:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
21198:     Procedure PSScriptCompile( Sender : TPSScript)
21199:     Procedure Compile1Click( Sender : TObject)
21200:     Procedure PSScriptExecute( Sender : TPSScript)
21201:     Procedure open1Click( Sender : TObject)
21202:     Procedure Save2Click( Sender : TObject)
21203:     Procedure Savebefore1Click( Sender : TObject)
21204:     Procedure Largefont1Click( Sender : TObject)
21205:     Procedure FormActivate( Sender : TObject)
21206:     Procedure SBytecode1Click( Sender : TObject)
21207:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
21208:     Procedure Saveas3Click( Sender : TObject)
21209:     Procedure Clear1Click( Sender : TObject)
21210:     Procedure Slinenumbers1Click( Sender : TObject)
21211:     Procedure About1Click( Sender : TObject)
21212:     Procedure Search1Click( Sender : TObject)
21213:     Procedure FormCreate( Sender : TObject)
21214:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
21215:                                                     var Action : TSynReplaceAction)
21216:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
21217:     Procedure WordWrap1Click( Sender : TObject)
```

```
21218:     Procedure SearchNext1Click( Sender : TObject)
21219:     Procedure Replace1Click( Sender : TObject)
21220:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
21221:     Procedure ShowInclude1Click( Sender : TObject)
21222:     Procedure Printout1Click( Sender : TObject)
21223:     Procedure mnuPrintFont1Click( Sender : TObject)
21224:     Procedure Include1Click( Sender : TObject)
21225:     Procedure FormDestroy( Sender : TObject)
21226:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
21227:     Procedure UpdateView1Click( Sender : TObject)
21228:     Procedure CodeCompletionList1Click( Sender : TObject)
21229:     Procedure SaveOutput1Click( Sender : TObject)
21230:     Procedure ExportClipboard1Click( Sender : TObject)
21231:     Procedure Close1Click( Sender : TObject)
21232:     Procedure Manual1Click( Sender : TObject)
21233:     Procedure LoadLastFile1Click( Sender : TObject)
21234:     Procedure Memo1Change( Sender : TObject)
21235:     Procedure Decompile1Click( Sender : TObject)
21236:     Procedure StepInto1Click( Sender : TObject)
21237:     Procedure StepOut1Click( Sender : TObject)
21238:     Procedure Reset1Click( Sender : TObject)
21239:     Procedure cedebugAfterExecute( Sender : TPSScript)
21240:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
21241:     Procedure cedebugCompile( Sender : TPSScript)
21242:     Procedure cedebugExecute( Sender : TPSScript)
21243:     Procedure cedebugIdle( Sender : TObject)
21244:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
21245:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
21246:     Procedure BreakPointMenuClick( Sender : TObject)
21247:     Procedure DebugRun1Click( Sender : TObject)
21248:     Procedure tutorial4Click( Sender : TObject)
21249:     Procedure ImportfromClipboard1Click( Sender : TObject)
21250:     Procedure ImportfromClipboard2Click( Sender : TObject)
21251:     Procedure tutorial1Click( Sender : TObject)
21252:     Procedure ShowSpecChars1Click( Sender : TObject)
21253:     Procedure StatusBar1DblClick( Sender : TObject)
21254:     Procedure PSScriptLine( Sender : TObject)
21255:     Procedure OpenDirectory1Click( Sender : TObject)
21256:     Procedure procMessClick( Sender : TObject)
21257:     Procedure tbtnUseCaseClick( Sender : TObject)
21258:     Procedure EditFont1Click( Sender : TObject)
21259:     Procedure tutorial21Click( Sender : TObject)
21260:     Procedure tutorial31Click( Sender : TObject)
21261:     Procedure HTMLSyntax1Click( Sender : TObject)
21262:     Procedure ShowInterfaces1Click( Sender : TObject)
21263:     Procedure Tutorial5Click( Sender : TObject)
21264:     Procedure ShowLastException1Click( Sender : TObject)
21265:     Procedure PlayMP31Click( Sender : TObject)
21266:     Procedure AllFunctionsList1Click( Sender : TObject)
21267:     Procedure texSyntax1Click( Sender : TObject)
21268:     Procedure GetEMails1Click( Sender : TObject)
21269:     procedure DelphiSite1Click(Sender: TObject);
21270:     procedure TerminalStyle1Click(Sender: TObject);
21271:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
21272:     procedure ShellStyle1Click(Sender: TObject);
21273:     procedure Console1Click(Sender: TObject);     //3.2
21274:     procedure BigScreen1Click(Sender: TObject);
21275:     procedure Tutorial91Click(Sender: TObject);
21276:     procedure SaveScreenshotClick(Sender: TObject);
21277:     procedure Tutorial101Click(Sender: TObject);
21278:     procedure SQLSyntax1Click(Sender: TObject);
21279:     procedure XMLSyntax1Click(Sender: TObject);
21280:     procedure ComponentCount1Click(Sender: TObject);
21281:     procedure NewInstance1Click(Sender: TObject);
21282:     procedure CSyntax1Click(Sender: TObject);
21283:     procedure Tutorial6Click(Sender: TObject);
21284:     procedure New1Click(Sender: TObject);
21285:     procedure AllObjectsList1Click(Sender: TObject);
21286:     procedure LoadBytecode1Click(Sender: TObject);
21287:     procedure CipherFile1Click(Sender: TObject);  //V3.5
21288:     procedure NewInstance1Click(Sender: TObject);
21289:     procedure toolbtnTutorialClick(Sender: TObject);
21290:     procedure Memory1Click(Sender: TObject);
21291:     procedure JavaSyntax1Click(Sender: TObject);
21292:     procedure SyntaxCheck1Click(Sender: TObject);
21293:     procedure ScriptExplorer1Click(Sender: TObject);
21294:     procedure FormOutput1Click(Sender: TObject);  //V3.6
21295:     procedure GotoEnd1Click(Sender: TObject);
21296:     procedure AllResourceList1Click(Sender: TObject);
21297:     procedure tbtn6resClick(Sender: TObject);   //V3.7
21298:     procedure Info1Click(Sender: TObject);
21299:     procedure Tutorial10Statistics1Click(Sender: TObject);
21300:     procedure Tutorial11Forms1Click(Sender: TObject);
21301:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
21302:     procedure ResourceExplore1Click(Sender: TObject);
21303:     procedure Info1Click(Sender: TObject);
21304:     procedure CryptoBox1Click(Sender: TObject);
21305:     procedure ModulesCount1Click(Sender: TObject);
21306:     procedure N4GewinntGame1Click(Sender: TObject);
```

```
21307:     procedure PHPSyntax1Click(Sender: TObject);
21308:     procedure SerialRS2321Click(Sender: TObject);
21309:     procedure CSyntax2Click(Sender: TObject);
21310:     procedure Calculator1Click(Sender: TObject);
21311:     procedure Tutorial13Ciphering1Click(Sender: TObject);
21312:     procedure Tutorial14Async1Click(Sender: TObject);
21313:     procedure PHPSyntax1Click(Sender: TObject);
21314:     procedure BtnZoomPlusClick(Sender: TObject);
21315:     procedure BtnZoomMinusClick(Sender: TObject);
21316:     procedure btnClassReportClick(Sender: TObject);
21317:     procedure ThreadDemo1Click(Sender: TObject);
21318:     procedure HEXView1Click(Sender: TObject);
21319:     procedure ExporttoHTML1Click(Sender: TObject);
21320:     procedure Minesweeper1Click(Sender: TObject);
21321:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
21322:     procedure sbvclhelpClick(Sender: TObject);
21323:     procedure DependencyWalker1Click(Sender: TObject);
21324:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
21325:     procedure WebScanner1Click(Sender: TObject);
21326:     procedure mnToolbar1Click(Sender: TObject);
21327:     procedure mnStatusbar2Click(Sender: TObject);
21328:     procedure mnConsole2Click(Sender: TObject);
21329:     procedure mnCoolbar2Click(Sender: TObject);
21330:     procedure mnSplitter2Click(Sender: TObject);
21331:     procedure WebServer1Click(Sender: TObject);
21332:     procedure PerlSyntax1Click(Sender: TObject);
21333:     procedure PythonSyntax1Click(Sender: TObject);
21334:     procedure DMathLibrary1Click(Sender: TObject);
21335:     procedure IntfNavigator1Click(Sender: TObject);
21336:     procedure FullTextFinder1Click(Sender: TObject);
21337:     function AppName: string;
21338:     function ScriptName: string;
21339:     function LastName: string;
21340:     procedure FractalDemo1Click(Sender: TObject);
21341:     procedure SimuLogBox1Click(Sender: TObject);
21342:     procedure OpenExamples1Click(Sender: TObject);
21343:     procedure Halt1Click(Sender: TObject);
21344:     procedure Stop;
21345:     procedure CodeSearch1Click(Sender: TObject);
21346:     procedure RubySyntax1Click(Sender: TObject);
21347:     procedure Undo1Click(Sender: TObject);
21348:     procedure LinuxShellScript1Click(Sender: TObject);
21349:     procedure WebScannerDirect(urls: string);
21350:     procedure WebScanner(urls: string);
21351:     procedure LoadInterfaceList2;
21352:     procedure DLLSpy1Click(Sender: TObject);
21353:     procedure Memo1DblClick(Sender: TObject);
21354:     procedure URILinksClicks1Click(Sender: TObject);
21355:     procedure GotoLine1Click(Sender: TObject);
21356:     procedure ConfigFile1Click(Sender: TObject);
21357:     Procedure Sort1IntflistClick( Sender : TObject)
21358:     Procedure Redo1Click( Sender : TObject)
21359:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
21360:     Procedure IndentSelection1Click( Sender : TObject)
21361:     Procedure UnindentSection1Click( Sender : TObject)
21362:     Procedure SkyStyle1Click( Sender : TObject)
21363:     Procedure CountWords1Click( Sender : TObject)
21364:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
21365:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
21366:     Procedure Bookmark11Click( Sender : TObject)
21367:     Procedure Bookmark21Click( Sender : TObject)
21368:     Procedure Bookmark31Click( Sender : TObject)
21369:     Procedure Bookmark41Click( Sender : TObject)
21370:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
21371:     'STATMemoryReport', 'boolean', iptrw);
21372:     'IPPort', 'integer', iptrw);
21373:     'COMPort', 'integer', iptrw);
21374:     'lbintflist', 'TListBox', iptrw);
21375:     Function GetStatChange : boolean
21376:     Procedure SetStatChange( vstat : boolean)
21377:     Function GetActFileName : string
21378:     Procedure SetActFileName( vname : string)
21379:     Function GetLastFileName : string
21380:     Procedure SetLastFileName( vname : string)
21381:     Procedure WebScannerDirect( urls : string)
21382:     Procedure LoadInterfaceList2
21383:     Function GetStatExecuteShell : boolean
21384:     Procedure DoEditorExecuteCommand( EditorCommand : word)
21385:     function GetActiveLineColor: TColor
21386:     procedure SetActiveLineColor(acolor: TColor)
21387:     procedure ScriptListbox1Click(Sender: TObject);
21388:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
21389:     procedure EnlargeGutter1Click(Sender: TObject);
21390:     procedure Tetris1Click(Sender: TObject);
21391:     procedure ToDoList1Click(Sender: TObject);
21392:     procedure ProcessList1Click(Sender: TObject);
21393:     procedure MetricReport1Click(Sender: TObject);
21394:     procedure ProcessList1Click(Sender: TObject);
21395:     procedure TCPSockets1Click(Sender: TObject);
```

```
21396:     procedure ConfigUpdate1Click(Sender: TObject);
21397:     procedure ADOWorkbench1Click(Sender: TObject);
21398:     procedure SocketServer1Click(Sender: TObject);
21399:     procedure FormDemo1Click(Sender: TObject);
21400:     procedure Richedit1Click(Sender: TObject);
21401:     procedure SimpleBrowser1Click(Sender: TObject);
21402:     procedure DOSShell1Click(Sender: TObject);
21403:     procedure SynExport1Click(Sender: TObject);
21404:     procedure ExporttoRTF1Click(Sender: TObject);
21405:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
21406:     procedure SOAPTester1Click(Sender: TObject);
21407:     procedure Sniffer1Click(Sender: TObject);
21408:     procedure AutoDetectSyntax1Click(Sender: TObject);
21409:     procedure FPlot1Click(Sender: TObject);
21410:     procedure PasStyle1Click(Sender: TObject);
21411:     procedure Tutorial183RGBLED1Click(Sender: TObject);
21412:     procedure Reversi1Click(Sender: TObject);
21413:     procedure ManualmaXbox1Click(Sender: TObject);
21414:     procedure BlaisePascalMagazine1Click(Sender: TObject);
21415:     procedure AddToDo1Click(Sender: TObject);
21416:     procedure CreateGUID1Click(Sender: TObject);
21417:     procedure Tutorial27XML1Click(Sender: TObject);
21418:     procedure CreateDLLStub1Click(Sender: TObject);
21419:     procedure Tutorial28DLL1Click(Sender: TObject);');
21420:     procedure ResetKeyPressed;');
21421:     procedure KeyPressedFalse;
21422:     procedure FileChanges1Click(Sender: TObject);');
21423:     procedure OpenGLTry1Click(Sender: TObject);');
21424:     procedure AllUnitList1Click(Sender: TObject);');
21425:     procedure Tutorial29UMLClick(Sender: TObject);
21426:     procedure CreateHeader1Click(Sender: TObject);
21427:     procedure Oscilloscope1Click(Sender: TObject);');
21428:     procedure Tutorial30WOT1Click(Sender: TObject);');
21429:     procedure GetWebScript1Click(Sender: TObject);');
21430:     procedure Checkers1Click(Sender: TObject);');
21431:     procedure TaskMgr1Click(Sender: TObject);');
21432:     procedure WebCam1Click(Sender: TObject);');
21433:     procedure Tutorial31Closure1Click(Sender: TObject);');
21434:     procedure GEOMapView1Click(Sender: TObject);');
21435:     procedure Run1Click(Sender: TObject);
21436:     MaxForm1.GPSSatView1Click, 'GPSSatView1Click');
21437:     MaxForm1.N3DLab1Click, 'N3DLab1Click');
21438:     procedure ExternalApp1Click(Sender: TObject);');
21439:     procedure PANView1Click(Sender: TObject);
21440:     procedure Tutorial39GEOMaps1Click(Sender: TObject);
21441:     procedure UnitConverter1Click(Sender: TObject);
21442: //----------------------------------------------------------------------------
21443: //**************mX4 Editor SynEdit Tools API ********************************
21444: //----------------------------------------------------------------------------
21445: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
21446: begin  //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
21447:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
21448:     Constructor Create( AOwner : TComponent)
21449:     SelStart', 'Integer', iptrw);
21450:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
21451:     Procedure UpdateCaret
21452:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
21453:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
21454:     Procedure BeginUndoBlock
21455:     Procedure BeginUpdate
21456:     Function CaretInView : Boolean
21457:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
21458:     Procedure Clear
21459:     Procedure ClearAll
21460:     Procedure ClearBookMark( BookMark : Integer)
21461:     Procedure ClearSelection
21462:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
21463:     Procedure ClearUndo
21464:     Procedure CopyToClipboard
21465:     Procedure CutToClipboard
21466:     Procedure DoCopyToClipboard( const SText : string)
21467:     Procedure EndUndoBlock
21468:     Procedure EndUpdate
21469:     Procedure EnsureCursorPosVisible
21470:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
21471:     Procedure FindMatchingBracket
21472:     Function GetMatchingBracket : TBufferCoord
21473:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
21474:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
21475:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
21476:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
21477:               : TSynHighlighterAttributes) : boolean
21478:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
21479:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
21480:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
21481:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
21482:     Procedure GotoBookMark( BookMark : Integer)
21483:     Procedure GotoLineAndCenter( ALine : Integer)
21484:     Function IdentChars : TSynIdentChars
```

```
21485:     Procedure InvalidateGutter
21486:     Procedure InvalidateGutterLine( aLine : integer)
21487:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
21488:     Procedure InvalidateLine( Line : integer)
21489:     Procedure InvalidateLines( FirstLine, LastLine : integer)
21490:     Procedure InvalidateSelection
21491:     Function IsBookmark( BookMark : integer) : boolean
21492:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
21493:     Procedure LockUndo
21494:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
21495:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
21496:     Function LineToRow( aLine : integer) : integer
21497:     Function RowToLine( aRow : integer) : integer
21498:     Function NextWordPos : TBufferCoord
21499:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
21500:     Procedure PasteFromClipboard
21501:     Function WordStart : TBufferCoord
21502:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
21503:     Function WordEnd : TBufferCoord
21504:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
21505:     Function PrevWordPos : TBufferCoord
21506:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
21507:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
21508:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
21509:     Procedure Redo
21510:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
21511:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
21512:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
21513:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
21514:     Procedure SelectAll
21515:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
21516:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
21517:     Procedure SetDefaultKeystrokes
21518:     Procedure SetSelWord
21519:     Procedure SetWordBlock( Value : TBufferCoord)
21520:     Procedure Undo
21521:     Procedure UnlockUndo
21522:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
21523:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
21524:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
21525:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
21526:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
21527:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
21528:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
21529:     Procedure AddFocusControl( aControl : TWinControl)
21530:     Procedure RemoveFocusControl( aControl : TWinControl)
21531:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
21532:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
21533:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
21534:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
21535:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
21536:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
21537:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
21538:     Procedure RemoveLinesPointer
21539:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
21540:     Procedure UnHookTextBuffer
21541:     BlockBegin', 'TBufferCoord', iptrw);
21542:     BlockEnd', 'TBufferCoord', iptrw);
21543:     CanPaste', 'Boolean', iptr);
21544:     CanRedo', 'boolean', iptr);
21545:     CanUndo', 'boolean', iptr);
21546:     CaretX', 'Integer', iptrw);
21547:     CaretY', 'Integer', iptrw);
21548:     CaretXY', 'TBufferCoord', iptrw);
21549:     ActiveLineColor', 'TColor', iptrw);
21550:     DisplayX', 'Integer', iptr);
21551:     DisplayY', 'Integer', iptr);
21552:     DisplayXY', 'TDisplayCoord', iptr);
21553:     DisplayLineCount', 'integer', iptr);
21554:     CharsInWindow', 'Integer', iptr);
21555:     CharWidth', 'integer', iptr);
21556:     Font', 'TFont', iptrw);
21557:     GutterWidth', 'Integer', iptr);
21558:     Highlighter', 'TSynCustomHighlighter', iptrw);
21559:     LeftChar', 'Integer', iptrw);
21560:     LineHeight', 'integer', iptr);
21561:     LinesInWindow', 'Integer', iptr);
21562:     LineText', 'string', iptrw);
21563:     Lines', 'TStrings', iptrw);
21564:     Marks', 'TSynEditMarkList', iptr);
21565:     MaxScrollWidth', 'integer', iptrw);
21566:     Modified', 'Boolean', iptrw);
21567:     PaintLock', 'Integer', iptr);
21568:     ReadOnly', 'Boolean', iptrw);
21569:     SearchEngine', 'TSynEditSearchCustom', iptrw);
21570:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
21571:     SelTabBlock', 'Boolean', iptr);
21572:     SelTabLine', 'Boolean', iptr);
21573:     SelText', 'string', iptrw);
```

```
21574:     StateFlags', 'TSynStateFlags', iptr);
21575:     Text', 'string', iptrw); TopLine', 'Integer', iptrw);
21576:     WordAtCursor', 'string', iptr);
21577:     WordAtMouse', 'string', iptr);
21578:     UndoList', 'TSynEditUndoList', iptr);
21579:     RedoList', 'TSynEditUndoList', iptr);
21580:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
21581:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
21582:     BorderStyle', 'TSynBorderStyle', iptrw);
21583:     ExtraLineSpacing', 'integer', iptrw);
21584:     Gutter', 'TSynGutter', iptrw);
21585:     HideSelection', 'boolean', iptrw);
21586:     InsertCaret', 'TSynEditCaretType', iptrw);
21587:     InsertMode', 'boolean', iptrw);
21588:     IsScrolling', 'Boolean', iptr);
21589:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
21590:     MaxUndo', 'Integer', iptrw);
21591:     Options', 'TSynEditorOptions', iptrw);
21592:     OverwriteCaret', 'TSynEditCaretType', iptrw);
21593:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
21594:     ScrollHintColor', 'TColor', iptrw);
21595:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
21596:     ScrollBars', 'TScrollStyle', iptrw);
21597:     SelectedColor', 'TSynSelectedColor', iptrw);
21598:     SelectionMode', 'TSynSelectionMode', iptrw);
21599:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
21600:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
21601:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
21602:     WordWrapGlyph', 'TSynGlyph', iptrw);
21603:     OnChange', 'TNotifyEvent', iptrw);
21604:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
21605:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
21606:     OnContextHelp', 'TContextHelpEvent', iptrw);
21607:     OnDropFiles', 'TDropFilesEvent', iptrw);
21608:     OnGutterClick', 'TGutterClickEvent', iptrw);
21609:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
21610:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
21611:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
21612:     OnPaint', 'TPaintEvent', iptrw);
21613:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
21614:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
21615:     OnReplaceText', 'TReplaceTextEvent', iptrw);
21616:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
21617:     OnStatusChange', 'TStatusChangeEvent', iptrw);
21618:     OnPaintTransient', 'TPaintTransient', iptrw);
21619:     OnScroll', 'TScrollEvent', iptrw);
21620:   end;
21621:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
21622:  Function GetPlaceableHighlighters : TSynHighlighterList
21623:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
21624:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
21625:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
21626:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
21627:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
21628:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
21629:  Function ConvertCodeStringToExtended( AString : String) : String
21630:  Function ConvertExtendedToCodeString( AString : String) : String
21631:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
21632:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
21633:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
21634:
21635:   TSynEditorOption = (
21636:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
21637:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
21638:                                // preceding line
21639:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
21640:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
21641:                                //direction any more
21642:     eoDragDropEditing,         //Allows to select a textblock and drag it in document to another location
21643:     eoDropFiles,               //Allows the editor accept OLE file drops
21644:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
21645:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
21646:     eoGroupUndo,               //When undoing/redoing actions,handle all cont.changes same kind in onecall
21647:                                //instead undoing/redoing each command separately
21648:     eoHalfPageScroll,          //By scrolling with page-up/page-down commands,only scroll half page attime
21649:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
21650:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
21651:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
21652:     eoNoCaret,                 //Makes it so the caret is never visible
21653:     eoNoSelection,             //Disables selecting text
21654:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
21655:     eoScrollByOneLess,         //Forces scrolling to be one less
21656:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
21657:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
21658:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
21659:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
21660:     eoShowSpecialChars,        //Shows the special Characters
21661:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
21662:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
```

```
21663:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
21664:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
21665:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
21666:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
21667:
21668:     ********************************Important Editor Short Cuts*****************************);
21669:     Double click to select a word and count words with highlightning.
21670:     Triple click to select a line.
21671:     CTRL+SHIFT+click to extend a selection.
21672:     Drag with the ALT key down to select columns of text !!!
21673:     Drag and drop is supported.
21674:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
21675:     Type CTRL+A to select all.
21676:     Type CTRL+N to set a new line.
21677:     Type CTRL+T to delete a line or token. //Tokenizer
21678:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
21679:     Type CTRL+Shift+T to add ToDo in line and list.
21680:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
21681:     Type CTRL[0..9] to jump or get to bookmarks.
21682:     Type Home to position cursor at beginning of current line and End to position it at end of line.
21683:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
21684:     Page Up and Page Down work as expected.
21685:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
21686:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
21687:
21688: {$ Short Key Positions Ctrl<A-Z>: }
21689: def
21690:    <A> Select All
21691:    <B> Count Words
21692:    <C> Copy
21693:    <D> Internet Start
21694:    <E> Script List
21695:    <F> Find
21696:    <G> Goto
21697:    <H> Mark Line
21698:    <I> Interface List
21699:    <J> Code Completion
21700:    <K> Console
21701:    <L> Interface List Box
21702:    <M> Font Smaller -
21703:    <N> New Line
21704:    <O> Open File
21705:    <P> Font Larger +
21706:    <Q> Quit
21707:    <R> Replace
21708:    <S> Save!
21709:    <T> Delete Line
21710:    <U> Use Case Editor
21711:    <V> Paste
21712:    <W> URI Links
21713:    <X> Reserved for coding use internal
21714:    <Y> Delete Line
21715:    <Z> Undo
21716:
21717: ref F1 Help
21718:     F2 Syntax Check
21719:     F3 Search Next
21720:     F4 New Instance
21721:     F5 Line Mark /Breakpoint
21722:     F6 Goto End
21723:     F7 Debug Step Into
21724:     F8 Debug Step Out
21725:     F9 Compile
21726:     F10 Menu
21727:     F11 Word Count Highlight
21728:     F12 Reserved for coding use internal
21729:
21730:    AddRegisteredVariable('Application', 'TApplication');
21731:    AddRegisteredVariable('Screen', 'TScreen');
21732:    AddRegisteredVariable('Self', 'TForm');
21733:    AddRegisteredVariable('Memo1', 'TSynMemo');
21734:    AddRegisteredVariable('memo2', 'TMemo');
21735:    AddRegisteredVariable('maxForm1', 'TMaxform1');  //!!
21736:    AddRegisteredVariable('debugout', 'Tdebugoutput');  //!!
21737:    AddRegisteredVariable('hlog','THotlog');  //!!
21738:    AddRegisteredVariable( it ,integer');  //for closure!!
21739:    AddRegisteredVariable( sr ,string');  //for closure
21740:    AddRegisteredVariable( bt ,boolean');  //for closure
21741:    AddRegisteredVariable( ft ,double');  //for closure
21742:    AddRegisteredVariable( srlist ,TStringlist');  //for closures
21743:
21744:  def ReservedWords: array[0..86] of string =
21745:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
21746:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
21747:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
21748:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
21749:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
21750:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
21751:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
```

```
21752:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
21753:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
21754:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
21755:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
21756:    AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
21757: //-------------------------------------------------------------------------
21758: //*************End of mX4 Public  Tools API *********************************
21759: //-------------------------------------------------------------------------
21760: maXbox Internal Inventory
21761: Amount of Functions: 14075
21762: Amount of Procedures: 8641
21763: Amount of Constructors: 1401
21764: Totals of Calls: 24117
21765: SHA1: Win 3.9.9.160 8B4D7070BA40BDE17EEDBE78121BB8B474A1D6CF
21766:
21767: ************************************************************
21768:  Doc Short Manual with 50 Tips!
21769: ************************************************************
21770: - Install: just save your maxboxdef.ini before and then extract the zip file!
21771: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
21772: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21773: - Menu: With <Crtl><F3> you can search for code on examples
21774: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21775: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
21776: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21777:
21778: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21779: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
21780: - Context Menu: You can printout your scripts as a pdf-file or html-export
21781: - Context: You do have a context menu with the right mouse click
21782:
21783: - Menu: With the UseCase Editor you can convert graphic formats too.
21784: - Menu: On menu Options you find Addons as compiled scripts
21785: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
21786: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21787: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21788: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
21789:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
21790:
21791: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21792: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21793: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21794: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
21795: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
21796: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
21797:          to delete and Click and mark to drag a bookmark
21798: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21799: - IDE: A file info with system and script information you find in menu Program/Information
21800: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21801: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21802: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21803: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21804: - Editor: Set Bookmarks to check your work in app or code
21805: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21806: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
21807: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21808:
21809: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
21810: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
21811: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
21812: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21813: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21814: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
21815: - Code: if you cant run a function try the second one, for ex. Voice() - Voice2(), inc() - inc1()
21816: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21817: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21818: - IDE menu /Help/Tools/ open the Task Manager
21819:
21820: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21821: - Add on when no browser is available start /Options/Add ons/Easy Browser
21822: - Add on SOAP Tester with SOP POST File
21823: - Add on IP Protocol Sniffer with List View
21824: - Add on OpenGL mX Robot Demo for android
21825: - Add on Checkers Game, Add on Oscilloscope /View GEO Map View3
21826:
21827: - Menu: Help/Tools as a Tool Section with DOS Opener
21828: - Menu Editor: export the code as RTF File
21829: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21830: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21831: - Context: Auto Detect of Syntax depending on file extension
21832: - Code: some Windows API function start with w in the name like wGetAtomName();
21833: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21834: - IDE File Check with menu ..View/File Changes/...
21835: - Context: Create a Header with Create Header in Navigator List at right window
21836: - Code: use  SysErrorMessage to get a real Error Description, Ex.
21837:    RemoveDir('c:\NoSuchFolder'); writeln('System Error Message:'+ SysErrorMessage(GetLastError));
21838: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21839: - Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click on link
21840: - Editor: with <Ctrl+Alt+R> you can write in RTF format with RichEdit link
```

```
21841: - Menu: Check Help/Tools! you can use richedit, DOS Shell or Explorer
21842:
21843: - using DLL example in maXbox: //function: {*****************************************}
21844:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21845:                                  cb: DWORD): BOOL; //stdcall;;
21846:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
21847:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21848:      External  'OpenProcess@kernel32.dll stdcall';
21849:
21850: GCC Compile Ex Script
21851: procedure TFormMain_btnCompileClick(Sender: TObject);
21852: begin
21853:   AProcess:= TProcess.Create(Nil);
21854:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
21855:       + ' -o "' + OpenDialog2.FileName + '"';
21856:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21857:     AProcess.Execute;
21858:     Memo2.Lines.BeginUpdate;
21859:     Memo2.Lines.Clear;
21860:     Memo2.Lines.LoadFromStream(AProcess.Output);
21861:     Memo2.Lines.EndUpdate;
21862:   finally
21863:     AProcess.Free;
21864:   end;
21865: end;
21866:
21867: ref Stopwatch pattern snip
21868: Time1:= Time;
21869:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
21870:   if initAndStartBoard then
21871:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
21872:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
21873:   PrintF('%d %s',[Trunc((Time-Time1)*24),
21874:                   FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
21875:
21876:    POST git-receive-pack (chunked)
21877: Pushing to https://github.com/maxkleiner/maXbox3.git
21878: To https://github.com/maxkleiner/maXbox3.git f127d21..c6a98da  masterbox2 -> masterbox2
21879: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
21880:
21881: History Shell Hell - Walk the Talk
21882: PCT Precompile Technology , mX4 ScriptStudio
21883: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
21884: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21885: emax layers: system-package-component-unit-class-function-block
21886: new keywords def ref using maXCalcF UML: use case act class state seq pac comp dep - lib lab
21887: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21888: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21889: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21890: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21891: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21892: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21893: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21894: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN,ParaDice 3D Cube Polygraph, OCR
21895: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21896: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21897: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21898: Inno Install and Setup Routines
21899: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21900: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21901: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21902: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
21903: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
21904: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21905: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
21906: ByteCode2, IPUtils2, GEOCode, CGI-Powtils, GPS_2, External App, Unit Converter
21907:
21908: Ref:
21909: https://unibe-ch.academia.edu/MaxKleiner
21910: http://www.slideshare.net/maxkleiner1
21911: http://www.scribd.com/max_kleiner
21912: http://www.delphiforfun.org/Programs/Utilities/index.htm
21913: http://www.slideshare.net/maxkleiner1
21914: http://s3.amazonaws.com/PreviewLinks/22959.html
21915: http://www.softwareschule.ch/arduino_training.pdf
21916: http://www.jrsoftware.org/isinfo.php
21917: http://www.be-precision.com/products/precision-builder/express/
21918: http://www.blaisepascal.eu/
21919: http://www.delphibasics.co.uk/
21920: http://www.youtube.com/watch?v=av89HAbqAsI
21921: http://www.angelfire.com/hi5/delphizeus/modal.html
21922: http://www.retroarchive.org/garbo/pc/turbopas/index.html
21923: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21924: http://delphi.org/2014/01/every-android-api-for-delphi/
21925: https://en.wikipedia.org/wiki/User:Maxkleiner
21926: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
21927: https://bitbucket.org/max_kleiner/maxbox3
21928: https://bitbucket.org/max_kleiner/maxbox3/downloads
21929: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
```

```
21930:
21931: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
21932: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21933: =renderBasicSearchNarrative&q=%s';
21934: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
21935: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
21936:
21937: function OpenMap(const Data: string): boolean;
21938: var encURL: string;
21939: begin
21940:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
21941:   try  //HttpGet(EncodedURL, mapStream);   //WinInet
21942:    Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
21943:    //OpenDoc(Exepath+'openmapx.html');
21944:    S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
21945:   finally
21946:     encURL:= '';
21947:   end;
21948: end;
21949:
21950: procedure GetGEOMap(C_form,apath: string; const Data: string);
21951: var encodedURL: string; mapStream: TMemoryStream;
21952: begin
21953:   //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPEncode(Data)]);
21954:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
21955:   mapStream:= TMemoryStream.create;
21956:   try
21957:     Wininet_HttpGet(EncodedURL, mapStream);   //WinInet
21958:     mapStream.Position:= 0;
21959:     mapStream.Savetofile(apath); // OpenDoc(apath);
21960:     S_ShellExecute(apath,'',seCmdOpen);
21961:   finally
21962:     mapStream.Free;
21963:   end;
21964: end;
21965:
21966: Procedure BtnFactory(a,b,c,d:smallint; title,apic:string;
21967:                              var abtn:TBitBtn; anEvent:TNotifyEvent; afrm:TForm);
21968: begin
21969:   abtn:= TBitBtn.create(afrm);
21970:   with abtn do begin
21971:     parent:= afrm;
21972:     setBounds(a,b,c,d)
21973:     font.size:= 12;
21974:     glyph.LoadFromResourceName(HINSTANCE, apic);
21975:     mXButton(5,5,width, height,12,12,handle);
21976:     caption:= title;
21977:     onClick:= anEvent As TNotifyEvent;
21978:   end;
21979: end;
21980:
21981:  function  MySoundcard: Longint;
21982:     external 'waveOutGetNumDevs@winmm.dll stdcall';
21983: function isSound: boolean;
21984: begin result:= mySoundcard > 0 end;
21985:
21986: function ListIdentical2(l1,l2:TStringList): Boolean;
21987: begin
21988:   Result:= False;
21989:   if l1.count = l2.count then begin
21990:     for it:= 0 to l1.count-1 do
21991:       if (l1[it] <> l2[it]) then Exit;
21992:     Result:= True;
21993:   end;
21994: end;
21995:
21996:
21997:
21998: ************************************************************
21999:  unit List asm internal end
22000: ************************************************************
22001: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
22002: 02 unit SIRegister_IdStrings                  //Indy Sockets
22003: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
22004: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
22005: 05 unit IFSI_WinForm1puzzle;                   //maXbox
22006: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
22007: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
22008: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
22009: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
22010: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
22011: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
22012: 12 unit uPSCompiler.pas;                      //PS kernel functions
22013: 13 unit uPSI_DBCommon;                        //DB Common_Routines and Types
22014: 14 unit uPSI_Printers.pas                     //Delphi VCL
22015: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
22016: 16 unit uPSC_comobj;                         //COM Functions
22017: 17 unit uPSI_Clipbrd;                        //Delphi VCL
22018: 18 unit Filectrl in IFSI_SysUtils_max;       //VCL Runtime
```

```
22019: 19 unit uPSI_SqlExpr;                    //DBX3
22020: 20 unit uPSI_ADODB;                      //ADODB
22021: 21 unit uPSI_StrHlpr;                    //String Helper Routines
22022: 22 unit uPSI_DateUtils;                  //Expansion to DateTimeLib
22023: 23 unit uPSI_FileUtils;                  //Expansion to Sys/File Utils
22024: 24 unit JUtils / gsUtils;                //Jedi / Metabase
22025: 25 unit JvFunctions_max;                 //Jedi Functions
22026: 26 unit HTTPParser;                      //Delphi VCL
22027: 27 unit HTTPUtil;                       //Delphi VCL
22028: 28 unit uPSI_XMLUtil;                   //Delphi VCL
22029: 29 unit uPSI_SOAPHTTPClient;            //Delphi VCL SOAP WebService V3.5
22030: 30 unit uPSI_Contnrs;                   //Delphi RTL Container of Classes
22031: 31 unit uPSI_MaskUtils;                 //RTL Edit and Mask functions
22032: 32 unit uPSI_MyBigInt;                  //big integer class with Math
22033: 33 unit uPSI_ConvUtils;                 //Delphi VCL Conversions engine
22034: 34 unit Types_Variants;                 //Delphi\Win32\rtl\sys
22035: 35 unit uPSI_IdHashSHA1;               //Indy Crypto Lib
22036: 36 unit uPSI_IdHashMessageDigest        //Indy Crypto;
22037: 37 unit uPSI_IdASN1Util;                //Indy ASN1Utility Routines;
22038: 38 unit uPSI_IdLogFile;                 //Indy Logger from LogBase
22039: 39 unit uPSI_IdIcmpClient;              //Indy Ping ICMP
22040: 40 unit uPSI_IdHashMessageDigest_max    //Indy Crypto &OpenSSL;
22041: 41 unit uPSI_FileCtrl;                  //Delphi RTL
22042: 42 unit uPSI_Outline;                   //Delphi VCL
22043: 43 unit uPSI_ScktComp;                  //Delphi RTL
22044: 44 unit uPSI_Calendar;                  //Delphi VCL
22045: 45 unit uPSI_VListView                  //VListView;
22046: 46 unit uPSI_DBGrids;                   //Delphi VCL
22047: 47 unit uPSI_DBCtrls;                   //Delphi VCL
22048: 48 unit ide_debugoutput;               //maXbox
22049: 49 unit uPSI_ComCtrls;                 //Delphi VCL
22050: 50 unit uPSC_stdctrls+;                //Delphi VCL
22051: 51 unit uPSI_Dialogs;                   //Delphi VCL
22052: 52 unit uPSI_StdConvs;                 //Delphi RTL
22053: 53 unit uPSI_DBClient;                 //Delphi RTL
22054: 54 unit uPSI_DBPlatform;               //Delphi RTL
22055: 55 unit uPSI_Provider;                 //Delphi RTL
22056: 56 unit uPSI_FMTBcd;                   //Delphi RTL
22057: 57 unit uPSI_DBCGrids;                 //Delphi VCL
22058: 58 unit uPSI_CDSUtil;                  //MIDAS
22059: 59 unit uPSI_VarHlpr;                  //Delphi RTL
22060: 60 unit uPSI_ExtDlgs;                  //Delphi VCL
22061: 61 unit sdpStopwatch;                  //maXbox
22062: 62 unit uPSI_JclStatistics;            //JCL
22063: 63 unit uPSI_JclLogic;                 //JCL
22064: 64 unit uPSI_JclMiscel;                //JCL
22065: 65 unit uPSI_JclMath_max;              //JCL RTL
22066: 66 unit uPSI_uTPLb_StreamUtils;        //LockBox 3
22067: 67 unit uPSI_MathUtils;                //BCB
22068: 68 unit uPSI_JclMultimedia;            //JCL
22069: 69 unit uPSI_WideStrUtils;             //Delphi API/RTL
22070: 70 unit uPSI_GraphUtil;                //Delphi RTL
22071: 71 unit uPSI_TypeTrans;                //Delphi RTL
22072: 72 unit uPSI_HTTPApp;                  //Delphi VCL
22073: 73 unit uPSI_DBWeb;                    //Delphi VCL
22074: 74 unit uPSI_DBBdeWeb;                 //Delphi VCL
22075: 75 unit uPSI_DBXpressWeb;              //Delphi VCL
22076: 76 unit uPSI_ShadowWnd;                //Delphi VCL
22077: 77 unit uPSI_ToolWin;                  //Delphi VCL
22078: 78 unit uPSI_Tabs;                     //Delphi VCL
22079: 79 unit uPSI_JclGraphUtils;            //JCL
22080: 80 unit uPSI_JclCounter;               //JCL
22081: 81 unit uPSI_JclSysInfo;               //JCL
22082: 82 unit uPSI_JclSecurity;              //JCL
22083: 83 unit uPSI_JclFileUtils;             //JCL
22084: 84 unit uPSI_IdUserAccounts;           //Indy
22085: 85 unit uPSI_IdAuthentication;         //Indy
22086: 86 unit uPSI_uTPLb_AES;                //LockBox 3
22087: 87 unit uPSI_IdHashSHA1;               //LockBox 3
22088: 88 unit uTPLb_BlockCipher;             //LockBox 3
22089: 89 unit uPSI_ValEdit.pas;              //Delphi VCL
22090: 90 unit uPSI_JvVCLUtils;               //JCL
22091: 91 unit uPSI_JvDBUtil;                 //JCL
22092: 92 unit uPSI_JvDBUtils;                //JCL
22093: 93 unit uPSI_JvAppUtils;               //JCL
22094: 94 unit uPSI_JvCtrlUtils;              //JCL
22095: 95 unit uPSI_JvFormToHtml;             //JCL
22096: 96 unit uPSI_JvParsing;                //JCL
22097: 97 unit uPSI_SerDlgs;                  //Toolbox
22098: 98 unit uPSI_Serial;                   //Toolbox
22099: 99 unit uPSI_JvComponent;              //JCL
22100: 100 unit uPSI_JvCalc;                  //JCL
22101: 101 unit uPSI_JvBdeUtils;              //JCL
22102: 102 unit uPSI_JvDateUtil;              //JCL
22103: 103 unit uPSI_JvGenetic;               //JCL
22104: 104 unit uPSI_JclBase;                 //JCL
22105: 105 unit uPSI_JvUtils;                 //JCL
22106: 106 unit uPSI_JvStrUtil;               //JCL
22107: 107 unit uPSI_JvStrUtils;              //JCL
```

```
22108: 108 unit uPSI_JvFileUtil;                    //JCL
22109: 109 unit uPSI_JvMemoryInfos;                 //JCL
22110: 110 unit uPSI_JvComputerInfo;                //JCL
22111: 111 unit uPSI_JvgCommClasses;                //JCL
22112: 112 unit uPSI_JvgLogics;                     //JCL
22113: 113 unit uPSI_JvLED;                         //JCL
22114: 114 unit uPSI_JvTurtle;                      //JCL
22115: 115 unit uPSI_SortThds; unit uPSI_ThSort;    //maXbox
22116: 116 unit uPSI_JvgUtils;                      //JCL
22117: 117 unit uPSI_JvExprParser;                  //JCL
22118: 118 unit uPSI_HexDump;                       //Borland
22119: 119 unit uPSI_DBLogDlg;                      //VCL
22120: 120 unit uPSI_SqlTimSt;                      //RTL
22121: 121 unit uPSI_JvHtmlParser;                  //JCL
22122: 122 unit uPSI_JvgXMLSerializer;             //JCL
22123: 123 unit uPSI_JvJCLUtils;                    //JCL
22124: 124 unit uPSI_JvStrings;                     //JCL
22125: 125 unit uPSI_uTPLb_IntegerUtils;           //TurboPower
22126: 126 unit uPSI_uTPLb_HugeCardinal;           //TurboPower
22127: 127 unit uPSI_uTPLb_HugeCardinalUtils;      //TurboPower
22128: 128 unit uPSI_SynRegExpr;                    //SynEdit
22129: 129 unit uPSI_StUtils;                      //SysTools4
22130: 130 unit uPSI_StToHTML;                     //SysTools4
22131: 131 unit uPSI_StStrms;                      //SysTools4
22132: 132 unit uPSI_StFIN;                        //SysTools4
22133: 133 unit uPSI_StAstroP;                     //SysTools4
22134: 134 unit uPSI_StStat;                       //SysTools4
22135: 135 unit uPSI_StNetCon;                     //SysTools4
22136: 136 unit uPSI_StDecMth;                     //SysTools4
22137: 137 unit uPSI_StOStr;                       //SysTools4
22138: 138 unit uPSI_StPtrns;                      //SysTools4
22139: 139 unit uPSI_StNetMsg;                     //SysTools4
22140: 140 unit uPSI_StMath;                       //SysTools4
22141: 141 unit uPSI_StExpEng;                     //SysTools4
22142: 142 unit uPSI_StCRC;                        //SysTools4
22143: 143 unit uPSI_StExport,                     //SysTools4
22144: 144 unit uPSI_StExpLog,                     //SysTools4
22145: 145 unit uPSI_ActnList;                     //Delphi VCL
22146: 146 unit uPSI_jpeg;                        //Borland
22147: 147 unit uPSI_StRandom;                     //SysTools4
22148: 148 unit uPSI_StDict;                       //SysTools4
22149: 149 unit uPSI_StBCD;                        //SysTools4
22150: 150 unit uPSI_StTxtDat;                     //SysTools4
22151: 151 unit uPSI_StRegEx;                      //SysTools4
22152: 152 unit uPSI_IMouse;                       //VCL
22153: 153 unit uPSI_SyncObjs;                     //VCL
22154: 154 unit uPSI_AsyncCalls;                   //Hausladen
22155: 155 unit uPSI_ParallelJobs;                 //Saraiva
22156: 156 unit uPSI_Variants;                     //VCL
22157: 157 unit uPSI_VarCmplx;                     //VCL Wolfram
22158: 158 unit uPSI_DTDSchema;                    //VCL
22159: 159 unit uPSI_ShLwApi;                      //Brakel
22160: 160 unit uPSI_IBUtils;                      //VCL
22161: 161 unit uPSI_CheckLst;                     //VCL
22162: 162 unit uPSI_JvSimpleXml;                  //JCL
22163: 163 unit uPSI_JclSimpleXml;                 //JCL
22164: 164 unit uPSI_JvXmlDatabase;                //JCL
22165: 165 unit uPSI_JvMaxPixel;                   //JCL
22166: 166 unit uPSI_JvItemsSearchs;               //JCL
22167: 167 unit uPSI_StExpEng2;                    //SysTools4
22168: 168 unit uPSI_StGenLog;                     //SysTools4
22169: 169 unit uPSI_JvLogFile;                    //Jcl
22170: 170 unit uPSI_CPort;                        //ComPort Lib v4.11
22171: 171 unit uPSI_CPortCtl;                     //ComPort
22172: 172 unit uPSI_CPortEsc;                     //ComPort
22173: 173 unit BarCodeScaner;                     //ComPort
22174: 174 unit uPSI_JvGraph;                      //JCL
22175: 175 unit uPSI_JvComCtrls;                   //JCL
22176: 176 unit uPSI_GUITesting;                   //D Unit
22177: 177 unit uPSI_JvFindFiles;                  //JCL
22178: 178 unit uPSI_StSystem;                     //SysTools4
22179: 179 unit uPSI_JvKeyboardStates;             //JCL
22180: 180 unit uPSI_JvMail;                       //JCL
22181: 181 unit uPSI_JclConsole;                   //JCL
22182: 182 unit uPSI_JclLANMan;                    //JCL
22183: 183 unit uPSI_IdCustomHTTPServer;           //Indy
22184: 184 unit IdHTTPServer                       //Indy
22185: 185 unit uPSI_IdTCPServer;                  //Indy
22186: 186 unit uPSI_IdSocketHandle;               //Indy
22187: 187 unit uPSI_IdIOHandlerSocket;            //Indy
22188: 188 unit IdIOHandler;                       //Indy
22189: 189 unit uPSI_cutils;                       //Bloodshed
22190: 190 unit uPSI_BoldUtils;                    //boldsoft
22191: 191 unit uPSI_IdSimpleServer;               //Indy
22192: 192 unit uPSI_IdSSLOpenSSL;                 //Indy
22193: 193 unit uPSI_IdMultipartFormData;          //Indy
22194: 194 unit uPSI_SynURIOpener;                 //SynEdit
22195: 195 unit uPSI_PerlRegEx;                    //PCRE
22196: 196 unit uPSI_IdHeaderList;                 //Indy
```

```
22197: 197 unit uPSI_StFirst;                        //SysTools4
22198: 198 unit uPSI_JvCtrls;                        //JCL
22199: 199 unit uPSI_IdTrivialFTPBase;               //Indy
22200: 200 unit uPSI_IdTrivialFTP;                   //Indy
22201: 201 unit uPSI_IdUDPBase;                      //Indy
22202: 202 unit uPSI_IdUDPClient;                    //Indy
22203: 203 unit uPSI_utypes;                         //for DMath.DLL
22204: 204 unit uPSI_ShellAPI;                       //Borland
22205: 205 unit uPSI_IdRemoteCMDClient;              //Indy
22206: 206 unit uPSI_IdRemoteCMDServer;              //Indy
22207: 207 unit IdRexecServer;                       //Indy
22208: 208 unit IdRexec; (unit uPSI_IdRexec;)        //Indy
22209: 209 unit IdUDPServer;                         //Indy
22210: 210 unit IdTimeUDPServer;                     //Indy
22211: 211 unit IdTimeServer;                        //Indy
22212: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)  //Indy
22213: 213 unit uPSI_IdIPWatch;                      //Indy
22214: 214 unit uPSI_IdIrcServer;                    //Indy
22215: 215 unit uPSI_IdMessageCollection;           //Indy
22216: 216 unit uPSI_cPEM;                          //Fundamentals 4
22217: 217 unit uPSI_cFundamentUtils;               //Fundamentals 4
22218: 218 unit uPSI_uwinplot;                      //DMath
22219: 219 unit uPSI_xrtl_util_CPUUtils;           //ExtentedRTL
22220: 220 unit uPSI_GR32_System;                   //Graphics32
22221: 221 unit uPSI_cFileUtils;                    //Fundamentals 4
22222: 222 unit uPSI_cDateTime; (timemachine)      //Fundamentals 4
22223: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
22224: 224 unit uPSI_cRandom;                       //Fundamentals 4
22225: 225 unit uPSI_ueval;                         //DMath
22226: 226 unit uPSI_xrtl_net_URIUtils;            //ExtendedRTL
22227: 227 unit xrtl_net_URIUtils;                  //ExtendedRTL
22228: 228 unit uPSI_ufft;  (FFT)                   //DMath
22229: 229 unit uPSI_DBXChannel;                    //Delphi
22230: 230 unit uPSI_DBXIndyChannel;               //Delphi Indy
22231: 231 unit uPSI_xrtl_util_COMCat;             //ExtendedRTL
22232: 232 unit uPSI_xrtl_util_StrUtils;           //ExtendedRTL
22233: 233 unit uPSI_xrtl_util_VariantUtils;       //ExtendedRTL
22234: 234 unit uPSI_xrtl_util_FileUtils;          //ExtendedRTL
22235: 235 unit xrtl_util_Compat;                  //ExtendedRTL
22236: 236 unit uPSI_OleAuto;                      //Borland
22237: 237 unit uPSI_xrtl_util_COMUtils;           //ExtendedRTL
22238: 238 unit uPSI_CmAdmCtl;                     //Borland
22239: 239 unit uPSI_ValEdit2;                     //VCL
22240: 240 unit uPSI_GR32;  //Graphics32          //Graphics32
22241: 241 unit uPSI_GR32_Image;                   //Graphics32
22242: 242 unit uPSI_xrtl_util_TimeUtils;          //ExtendedRTL
22243: 243 unit uPSI_xrtl_util_TimeZone;           //ExtendedRTL
22244: 244 unit uPSI_xrtl_util_TimeStamp;          //ExtendedRTL
22245: 245 unit uPSI_xrtl_util_Map;                //ExtendedRTL
22246: 246 unit uPSI_xrtl_util_Set;                //ExtendedRTL
22247: 247 unit uPSI_CPortMonitor;                 //ComPort
22248: 248 unit uPSI_StIniStm;                     //SysTools4
22249: 249 unit uPSI_GR32_ExtImage;                //Graphics32
22250: 250 unit uPSI_GR32_OrdinalMaps;             //Graphics32
22251: 251 unit uPSI_GR32_Rasterizers;             //Graphics32
22252: 252 unit uPSI_xrtl_util_Exception;          //ExtendedRTL
22253: 253 unit uPSI_xrtl_util_Value;              //ExtendedRTL
22254: 254 unit uPSI_xrtl_util_Compare;            //ExtendedRTL
22255: 255 unit uPSI_FlatSB;                       //VCL
22256: 256 unit uPSI_JvAnalogClock;                //JCL
22257: 257 unit uPSI_JvAlarms;                     //JCL
22258: 258 unit uPSI_JvSQLS;                       //JCL
22259: 259 unit uPSI_JvDBSecur;                    //JCL
22260: 260 unit uPSI_JvDBQBE;                      //JCL
22261: 261 unit uPSI_JvStarfield;                  //JCL
22262: 262 unit uPSI_JVCLMiscal;                   //JCL
22263: 263 unit uPSI_JvProfiler32;                 //JCL
22264: 264 unit uPSI_JvDirectories,                //JCL
22265: 265 unit uPSI_JclSchedule,                  //JCL
22266: 266 unit uPSI_JclSvcCtrl,                   //JCL
22267: 267 unit uPSI_JvSoundControl,               //JCL
22268: 268 unit uPSI_JvBDESQLScript,               //JCL
22269: 269 unit uPSI_JvgDigits,                    //JCL>
22270: 270 unit uPSI_ImgList;                      //TCustomImageList
22271: 271 unit uPSI_JclMIDI;                      //JCL>
22272: 272 unit uPSI_JclWinMidi;                   //JCL>
22273: 273 unit uPSI_JclNTFS;                      //JCL>
22274: 274 unit uPSI_JclAppInst;                   //JCL>
22275: 275 unit uPSI_JvRle;                        //JCL>
22276: 276 unit uPSI_JvRas32;                      //JCL>
22277: 277 unit uPSI_JvImageDrawThread,            //JCL>
22278: 278 unit uPSI_JvImageWindow,                //JCL>
22279: 279 unit uPSI_JvTransparentForm;            //JCL>
22280: 280 unit uPSI_JvWinDialogs;                 //JCL>
22281: 281 unit uPSI_JvSimLogic,                   //JCL>
22282: 282 unit uPSI_JvSimIndicator,               //JCL>
22283: 283 unit uPSI_JvSimPID,                     //JCL>
22284: 284 unit uPSI_JvSimPIDLinker,               //JCL>
22285: 285 unit uPSI_IdRFCReply;                    //Indy
```

```
22286: 286 unit uPSI_IdIdent;                          //Indy
22287: 287 unit uPSI_IdIdentServer;                    //Indy
22288: 288 unit uPSI_JvPatchFile;                      //JCL
22289: 289 unit uPSI_StNetPfm;                         //SysTools4
22290: 290 unit uPSI_StNet;                            //SysTools4
22291: 291 unit uPSI_JclPeImage;                       //JCL
22292: 292 unit uPSI_JclPrint;                         //JCL
22293: 293 unit uPSI_JclMime;                         //JCL
22294: 294 unit uPSI_JvRichEdit;                       //JCL
22295: 295 unit uPSI_JvDBRichEd;                       //JCL
22296: 296 unit uPSI_JvDice;                          //JCL
22297: 297 unit uPSI_JvFloatEdit;                     //JCL 3.9.8
22298: 298 unit uPSI_JvDirFrm;                        //JCL
22299: 299 unit uPSI_JvDualList;                      //JCL
22300: 300 unit uPSI_JvSwitch;                        ////JCL
22301: 301 unit uPSI_JvTimerLst;                      ////JCL
22302: 302 unit uPSI_JvMemTable;                      //JCL
22303: 303 unit uPSI_JvObjStr;                        //JCL
22304: 304 unit uPSI_StLArr;                          //SysTools4
22305: 305 unit uPSI_StWmDCpy;                        //SysTools4
22306: 306 unit uPSI_StText;                          //SysTools4
22307: 307 unit uPSI_StNTLog;                         //SysTools4
22308: 308 unit uPSI_xrtl_math_Integer;               //ExtendedRTL
22309: 309 unit uPSI_JvImagPrvw;                      //JCL
22310: 310 unit uPSI_JvFormPatch;                     //JCL
22311: 311 unit uPSI_JvPicClip;                       //JCL
22312: 312 unit uPSI_JvDataConv;                      //JCL
22313: 313 unit uPSI_JvCpuUsage;                      //JCL
22314: 314 unit uPSI_JclUnitConv_mX2;                 //JCL
22315: 315 unit JvDualListForm;                       //JCL
22316: 316 unit uPSI_JvCpuUsage2;                     //JCL
22317: 317 unit uPSI_JvParserForm;                    //JCL
22318: 318 unit uPSI_JvJanTreeView;                   //JCL
22319: 319 unit uPSI_JvTransLED;                      //JCL
22320: 320 unit uPSI_JvPlaylist;                      //JCL
22321: 321 unit uPSI_JvFormAutoSize;                  //JCL
22322: 322 unit uPSI_JvYearGridEditForm;              //JCL
22323: 323 unit uPSI_JvMarkupCommon;                  //JCL
22324: 324 unit uPSI_JvChart;                         //JCL
22325: 325 unit uPSI_JvXPCore;                        //JCL
22326: 326 unit uPSI_JvXPCoreUtils;                   //JCL
22327: 327 unit uPSI_StatsClasses;                    //mX4
22328: 328 unit uPSI_ExtCtrls2;                       //VCL
22329: 329 unit uPSI_JvUrlGrabbers;                   //JCL
22330: 330 unit uPSI_JvXmlTree;                       //JCL
22331: 331 unit uPSI_JvWavePlayer;                    //JCL
22332: 332 unit uPSI_JvUnicodeCanvas;                 //JCL
22333: 333 unit uPSI_JvTFUtils;                       //JCL
22334: 334 unit uPSI_IdServerIOHandler;               //Indy
22335: 335 unit uPSI_IdServerIOHandlerSocket;         //Indy
22336: 336 unit uPSI_IdMessageCoder;                  //Indy
22337: 337 unit uPSI_IdMessageCoderMIME;              //Indy
22338: 338 unit uPSI_IdMIMETypes;                     //Indy
22339: 339 unit uPSI_JvConverter;                     //JCL
22340: 340 unit uPSI_JvCsvParse;                      //JCL
22341: 341 unit uPSI_umath;  unit uPSI_ugamma;        //DMath
22342: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
22343: 343 unit uPSI_JvDBGridExport;                  //JCL
22344: 344 unit uPSI_JvgExport;                       //JCL
22345: 345 unit uPSI_JvSerialMaker;                   //JCL
22346: 346 unit uPSI_JvWin32;                         //JCL
22347: 347 unit uPSI_JvPaintFX;                       //JCL
22348: 348 unit uPSI_JvOracleDataSet; (beta)          //JCL
22349: 349 unit uPSI_JvValidators; (preview)          //JCL
22350: 350 unit uPSI_JvNTEventLog;                    //JCL
22351: 351 unit uPSI_ShellZipTool;                    //mX4
22352: 352 unit uPSI_JvJoystick;                      //JCL
22353: 353 unit uPSI_JvMailSlots;                     //JCL
22354: 354 unit uPSI_JclComplex;                      //JCL
22355: 355 unit uPSI_SynPdf;                          //Synopse
22356: 356 unit uPSI_Registry;                        //VCL
22357: 357 unit uPSI_TlHelp32;                        //VCL
22358: 358 unit uPSI_JclRegistry;                     //JCL
22359: 359 unit uPSI_JvAirBrush;                      //JCL
22360: 360 unit uPSI_mORMotReport;                    //Synopse
22361: 361 unit uPSI_JclLocales;                      //JCL
22362: 362 unit uPSI_SynEdit;                         //SynEdit
22363: 363 unit uPSI_SynEditTypes;                    //SynEdit
22364: 364 unit uPSI_SynMacroRecorder;                //SynEdit
22365: 365 unit uPSI_LongIntList;                     //SynEdit
22366: 366 unit uPSI_devcutils;                       //DevC
22367: 367 unit uPSI_SynEditMiscClasses;              //SynEdit
22368: 368 unit uPSI_SynEditRegexSearch;              //SynEdit
22369: 369 unit uPSI_SynEditHighlighter;              //SynEdit
22370: 370 unit uPSI_SynHighlighterPas;               //SynEdit
22371: 371 unit uPSI_JvSearchFiles;                   //JCL
22372: 372 unit uPSI_SynHighlighterAny;               //Lazarus
22373: 373 unit uPSI_SynEditKeyCmds;                  //SynEdit
22374: 374 unit uPSI_SynEditMiscProcs,                //SynEdit
```

```
22375: 375 unit uPSI_SynEditKbdHandler               //SynEdit
22376: 376 unit uPSI_JvAppInst,                      //JCL
22377: 377 unit uPSI_JvAppEvent;                     //JCL
22378: 378 unit uPSI_JvAppCommand;                   //JCL
22379: 379 unit uPSI_JvAnimTitle;                    //JCL
22380: 380 unit uPSI_JvAnimatedImage;                //JCL
22381: 381 unit uPSI_SynEditExport;                  //SynEdit
22382: 382 unit uPSI_SynExportHTML;                  //SynEdit
22383: 383 unit uPSI_SynExportRTF;                   //SynEdit
22384: 384 unit uPSI_SynEditSearch;                  //SynEdit
22385: 385 unit uPSI_fMain_back                      //maXbox;
22386: 386 unit uPSI_JvZoom;                         //JCL
22387: 387 unit uPSI_PMrand;                         //PM
22388: 388 unit uPSI_JvSticker;                      //JCL
22389: 389 unit uPSI_XmlVerySimple;                  //mX4
22390: 390 unit uPSI_Services;                       //ExtPascal
22391: 391 unit uPSI_ExtPascalUtils;                 //ExtPascal
22392: 392 unit uPSI_SocketsDelphi;                  //ExtPascal
22393: 393 unit uPSI_StBarC;                         //SysTools
22394: 394 unit uPSI_StDbBarC;                       //SysTools
22395: 395 unit uPSI_StBarPN;                        //SysTools
22396: 396 unit uPSI_StDbPNBC;                       //SysTools
22397: 397 unit uPSI_StDb2DBC;                       //SysTools
22398: 398 unit uPSI_StMoney;                        //SysTools
22399: 399 unit uPSI_JvForth;                        //JCL
22400: 400 unit uPSI_RestRequest;                    //mX4
22401: 401 unit uPSI_HttpRESTConnectionIndy;         //mX4
22402: 402 unit uPSI_JvXmlDatabase; //update        //JCL
22403: 403 unit uPSI_StAstro;                        //SysTools
22404: 404 unit uPSI_StSort;                         //SysTools
22405: 405 unit uPSI_StDate;                         //SysTools
22406: 406 unit uPSI_StDateSt;                       //SysTools
22407: 407 unit uPSI_StBase;                         //SysTools
22408: 408 unit uPSI_StVInfo;                        //SysTools
22409: 409 unit uPSI_JvBrowseFolder;                 //JCL
22410: 410 unit uPSI_JvBoxProcs;                     //JCL
22411: 411 unit uPSI_urandom; (unit uranuvag;)       //DMath
22412: 412 unit uPSI_usimann; (unit ugenalg;)       //DMath
22413: 413 unit uPSI_JvHighlighter;                  //JCL
22414: 414 unit uPSI_Diff;                           //mX4
22415: 415 unit uPSI_SpringWinAPI;                   //DSpring
22416: 416 unit uPSI_StBits;                         //SysTools
22417: 417 unit uPSI_TomDBQue;                       //mX4
22418: 418 unit uPSI_MultilangTranslator;            //mX4
22419: 419 unit uPSI_HyperLabel;                     //mX4
22420: 420 unit uPSI_Starter;                        //mX4
22421: 421 unit uPSI_FileAssocs;                     //devC
22422: 422 unit uPSI_devFileMonitorX;                //devC
22423: 423 unit uPSI_devrun;                         //devC
22424: 424 unit uPSI_devExec;                        //devC
22425: 425 unit uPSI_oysUtils;                       //devC
22426: 426 unit uPSI_DosCommand;                     //devC
22427: 427 unit uPSI_CppTokenizer;                   //devC
22428: 428 unit uPSI_JvHLParser;                     //devC
22429: 429 unit uPSI_JclMapi;                        //JCL
22430: 430 unit uPSI_JclShell;                       //JCL
22431: 431 unit uPSI_JclCOM;                         //JCL
22432: 432 unit uPSI_GR32_Math;                      //Graphics32
22433: 433 unit uPSI_GR32_LowLevel;                  //Graphics32
22434: 434 unit uPSI_SimpleHl;                       //mX4
22435: 435 unit uPSI_GR32_Filters,                   //Graphics32
22436: 436 unit uPSI_GR32_VectorMaps;                //Graphics32
22437: 437 unit uPSI_cXMLFunctions;                  //Fundamentals 4
22438: 438 unit uPSI_JvTimer;                        //JCL
22439: 439 unit uPSI_cHTTPUtils;                     //Fundamentals 4
22440: 440 unit uPSI_cTLSUtils;                      //Fundamentals 4
22441: 441 unit uPSI_JclGraphics;                    //JCL
22442: 442 unit uPSI_JclSynch;                       //JCL
22443: 443 unit uPSI_IdTelnet;                       //Indy
22444: 444 unit uPSI_IdTelnetServer,                 //Indy
22445: 445 unit uPSI_IdEcho,                         //Indy
22446: 446 unit uPSI_IdEchoServer,                   //Indy
22447: 447 unit uPSI_IdEchoUDP,                      //Indy
22448: 448 unit uPSI_IdEchoUDPServer,               //Indy
22449: 449 unit uPSI_IdSocks,                       //Indy
22450: 450 unit uPSI_IdAntiFreezeBase;              //Indy
22451: 451 unit uPSI_IdHostnameServer;              //Indy
22452: 452 unit uPSI_IdTunnelCommon,                //Indy
22453: 453 unit uPSI_IdTunnelMaster,                //Indy
22454: 454 unit uPSI_IdTunnelSlave,                 //Indy
22455: 455 unit uPSI_IdRSH,                         //Indy
22456: 456 unit uPSI_IdRSHServer,                   //Indy
22457: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
22458: 458 unit uPSI_MapReader,                     //devC
22459: 459 unit uPSI_LibTar,                        //devC
22460: 460 unit uPSI_IdStack;                       //Indy
22461: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
22462: 462 unit uPSI_IdChargenServer;               //Indy
22463: 463 unit uPSI_IdFTPServer,                   //Indy
```

```
22464: 464 unit uPSI_IdException,                    //Indy
22465: 465 unit uPSI_utexplot;                       //DMath
22466: 466 unit uPSI_uwinstr;                        //DMath
22467: 467 unit uPSI_VarRecUtils;                    //devC
22468: 468 unit uPSI_JvStringListToHtml,             //JCL
22469: 469 unit uPSI_JvStringHolder,                 //JCL
22470: 470 unit uPSI_IdCoder;                        //Indy
22471: 471 unit uPSI_SynHighlighterDfm;              //Synedit
22472: 472 unit uHighlighterProcs; in 471            //Synedit
22473: 473 unit uPSI_LazFileUtils,                   //LCL
22474: 474 unit uPSI_IDECmdLine;                     //LCL
22475: 475 unit uPSI_lazMasks;                       //LCL
22476: 476 unit uPSI_ip_misc;                        //mX4
22477: 477 unit uPSI_Barcode;                        //LCL
22478: 478 unit uPSI_SimpleXML;                      //LCL
22479: 479 unit uPSI_JclIniFiles;                    //JCL
22480: 480 unit uPSI_D2XXUnit;  {$X-}                //FTDI
22481: 481 unit uPSI_JclDateTime;                    //JCL
22482: 482 unit uPSI_JclEDI;                        //JCL
22483: 483 unit uPSI_JclMiscel2;                    //JCL
22484: 484 unit uPSI_JclValidation;                 //JCL
22485: 485 unit uPSI_JclAnsiStrings; {-PString}     //JCL
22486: 486 unit uPSI_SynEditMiscProcs2;             //Synedit
22487: 487 unit uPSI_JclStreams;                    //JCL
22488: 488 unit uPSI_QRCode;                        //mX4
22489: 489 unit uPSI_BlockSocket;                   //ExtPascal
22490: 490 unit uPSI_Masks,Utils                    //VCL
22491: 491 unit uPSI_synautil;                      //Synapse!
22492: 492 unit uPSI_JclMath_Class;                 //JCL RTL
22493: 493 unit ugamdist; //Gamma function          //DMath
22494: 494 unit uibeta, ucorrel; //IBeta           //DMath
22495: 495 unit uPSI_SRMgr;                         //mX4
22496: 496 unit uPSI_HotLog;                        //mX4
22497: 497 unit uPSI_DebugBox;                      //mX4
22498: 498 unit uPSI_ustrings;                      //DMath
22499: 499 unit uPSI_uregtest;                      //DMath
22500: 500 unit uPSI_usimplex;                      //DMath
22501: 501 unit uPSI_uhyper;                        //DMath
22502: 502 unit uPSI_IdHL7;                         //Indy
22503: 503 unit uPSI_IdIPMCastBase,                 //Indy
22504: 504 unit uPSI_IdIPMCastServer;               //Indy
22505: 505 unit uPSI_IdIPMCastClient;               //Indy
22506: 506 unit uPSI_unlfit; //nlregression        //DMath
22507: 507 unit uPSI_IdRawHeaders;                  //Indy
22508: 508 unit uPSI_IdRawClient;                   //Indy
22509: 509 unit uPSI_IdRawFunctions;                //Indy
22510: 510 unit uPSI_IdTCPStream;                   //Indy
22511: 511 unit uPSI_IdSNPP;                        //Indy
22512: 512 unit uPSI_St2DBarC;                      //SysTools
22513: 513 unit uPSI_ImageWin;  //FTL              //VCL
22514: 514 unit uPSI_CustomDrawTreeView; //FTL     //VCL
22515: 515 unit uPSI_GraphWin;  //FTL              //VCL
22516: 516 unit uPSI_actionMain;  //FTL            //VCL
22517: 517 unit uPSI_StSpawn;                       //SysTools
22518: 518 unit uPSI_CtlPanel;                      //VCL
22519: 519 unit uPSI_IdLPR;                         //Indy
22520: 520 unit uPSI_SockRequestInterpreter;        //Indy
22521: 521 unit uPSI_ulambert;                      //DMath
22522: 522 unit uPSI_ucholesk;                      //DMath
22523: 523 unit uPSI_SimpleDS;                      //VCL
22524: 524 unit uPSI_DBXSqlScanner;                 //VCL
22525: 525 unit uPSI_DBXMetaDataUtil;               //VCL
22526: 526 unit uPSI_Chart;                         //TEE
22527: 527 unit uPSI_TeeProcs;                      //TEE
22528: 528 unit mXBDEUtils;                         //mX4
22529: 529 unit uPSI_MDIEdit;                       //VCL
22530: 530 unit uPSI_CopyPrsr;                      //VCL
22531: 531 unit uPSI_SockApp;                       //VCL
22532: 532 unit uPSI_AppEvnts;                      //VCL
22533: 533 unit uPSI_ExtActns;                      //VCL
22534: 534 unit uPSI_TeEngine;                      //TEE
22535: 535 unit uPSI_CoolMain; //browser           //VCL
22536: 536 unit uPSI_StCRC;                         //SysTools
22537: 537 unit uPSI_StDecMth2;                     //SysTools
22538: 538 unit uPSI_frmExportMain;                 //Synedit
22539: 539 unit uPSI_SynDBEdit;                     //Synedit
22540: 540 unit uPSI_SynEditWildcardSearch;         //Synedit
22541: 541 unit uPSI_BoldComUtils;                  //BOLD
22542: 542 unit uPSI_BoldIsoDateTime;               //BOLD
22543: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils   //BOLD
22544: 544 unit uPSI_BoldXMLRequests;               //BOLD
22545: 545 unit uPSI_BoldStringList;                //BOLD
22546: 546 unit uPSI_BoldFileHandler;               //BOLD
22547: 547 unit uPSI_BoldContainers;                //BOLD
22548: 548 unit uPSI_BoldQueryUserDlg;              //BOLD
22549: 549 unit uPSI_BoldWinINet;                   //BOLD
22550: 550 unit uPSI_BoldQueue;                     //BOLD
22551: 551 unit uPSI_JvPcx;                         //JCL
22552: 552 unit uPSI_IdWhois;                       //Indy
```

```
22553: 553 unit uPSI_IdWhoIsServer;                //Indy
22554: 554 unit uPSI_IdGopher;                     //Indy
22555: 555 unit uPSI_IdDateTimeStamp;             //Indy
22556: 556 unit uPSI_IdDayTimeServer;             //Indy
22557: 557 unit uPSI_IdDayTimeUDP;                 //Indy
22558: 558 unit uPSI_IdDayTimeUDPServer;           //Indy
22559: 559 unit uPSI_IdDICTServer;                 //Indy
22560: 560 unit uPSI_IdDiscardServer;             //Indy
22561: 561 unit uPSI_IdDiscardUDPServer;          //Indy
22562: 562 unit uPSI_IdMappedFTP;                  //Indy
22563: 563 unit uPSI_IdMappedPortTCP;              //Indy
22564: 564 unit uPSI_IdGopherServer;              //Indy
22565: 565 unit uPSI_IdQotdServer;                //Indy
22566: 566 unit uPSI_JvRgbToHtml;                 //JCL
22567: 567 unit uPSI_JvRemLog,                    //JCL
22568: 568 unit uPSI_JvSysComp;                   //JCL
22569: 569 unit uPSI_JvTMTL;                      //JCL
22570: 570 unit uPSI_JvWinampAPI;                 //JCL
22571: 571 unit uPSI_MSysUtils;                   //mX4
22572: 572 unit uPSI_ESBMaths;                    //ESB
22573: 573 unit uPSI_ESBMaths2;                   //ESB
22574: 574 unit uPSI_uLkJSON;                     //Lk
22575: 575 unit uPSI_ZURL;  //Zeos                //Zeos
22576: 576 unit uPSI_ZSysUtils;                   //Zeos
22577: 577 unit unaUtils internals                //UNA
22578: 578 unit uPSI_ZMatchPattern;               //Zeos
22579: 579 unit uPSI_ZClasses;                    //Zeos
22580: 580 unit uPSI_ZCollections;                //Zeos
22581: 581 unit uPSI_ZEncoding;                   //Zeos
22582: 582 unit uPSI_IdRawBase;                   //Indy
22583: 583 unit uPSI_IdNTLM;                      //Indy
22584: 584 unit uPSI_IdNNTP;                      //Indy
22585: 585 unit uPSI_usniffer; //PortScanForm     //mX4
22586: 586 unit uPSI_IdCoderMIME;                 //Indy
22587: 587 unit uPSI_IdCoderUUE;                  //Indy
22588: 588 unit uPSI_IdCoderXXE;                  //Indy
22589: 589 unit uPSI_IdCoder3to4;                 //Indy
22590: 590 unit uPSI_IdCookie;                    //Indy
22591: 591 unit uPSI_IdCookieManager;             //Indy
22592: 592 unit uPSI_WDosSocketUtils;             //WDos
22593: 593 unit uPSI_WDosPlcUtils;                //WDos
22594: 594 unit uPSI_WDosPorts;                   //WDos
22595: 595 unit uPSI_WDosResolvers;               //WDos
22596: 596 unit uPSI_WDosTimers;                  //WDos
22597: 597 unit uPSI_WDosPlcs;                    //WDos
22598: 598 unit uPSI_WDosPneumatics;              //WDos
22599: 599 unit uPSI_IdFingerServer;              //Indy
22600: 600 unit uPSI_IdDNSResolver;               //Indy
22601: 601 unit uPSI_IdHTTPWebBrokerBridge;       //Indy
22602: 602 unit uPSI_IdIntercept;                 //Indy
22603: 603 unit uPSI_IdIPMCastBase;               //Indy
22604: 604 unit uPSI_IdLogBase;                   //Indy
22605: 605 unit uPSI_IdIOHandlerStream;           //Indy
22606: 606 unit uPSI_IdMappedPortUDP;             //Indy
22607: 607 unit uPSI_IdQOTDUDPServer;             //Indy
22608: 608 unit uPSI_IdQOTDUDP;                   //Indy
22609: 609 unit uPSI_IdSysLog;                    //Indy
22610: 610 unit uPSI_IdSysLogServer;              //Indy
22611: 611 unit uPSI_IdSysLogMessage;             //Indy
22612: 612 unit uPSI_IdTimeServer;                //Indy
22613: 613 unit uPSI_IdTimeUDP;                   //Indy
22614: 614 unit uPSI_IdTimeUDPServer;             //Indy
22615: 615 unit uPSI_IdUserAccounts;              //Indy
22616: 616 unit uPSI_TextUtils;                   //mX4
22617: 617 unit uPSI_MandelbrotEngine;            //mX4
22618: 618 unit uPSI_delphi_arduino_Unit1;        //mX4
22619: 619 unit uPSI_DTDSchema2;                  //mX4
22620: 620 unit uPSI_fplotMain;                   //DMath
22621: 621 unit uPSI_FindFileIter;                //mX4
22622: 622 unit uPSI_PppState;  (JclStrHashMap)   //PPP
22623: 623 unit uPSI_PppParser;                   //PPP
22624: 624 unit uPSI_PppLexer;                    //PPP
22625: 625 unit uPSI_PCharUtils;                  //PPP
22626: 626 unit uPSI_uJSON;                       //WU
22627: 627 unit uPSI_JclStrHashMap;               //JCL
22628: 628 unit uPSI_JclHookExcept;               //JCL
22629: 629 unit uPSI_EncdDecd;                    //VCL
22630: 630 unit uPSI_SockAppReg;                  //VCL
22631: 631 unit uPSI_PJFileHandle;                //PJ
22632: 632 unit uPSI_PJEnvVars;                   //PJ
22633: 633 unit uPSI_PJPipe;                      //PJ
22634: 634 unit uPSI_PJPipeFilters;               //PJ
22635: 635 unit uPSI_PJConsoleApp;                //PJ
22636: 636 unit uPSI_UConsoleAppEx;               //PJ
22637: 637 unit uPSI_DbxSocketChannelNative,      //VCL
22638: 638 unit uPSI_DbxDataGenerator,            //VCL
22639: 639 unit uPSI_DBXClient;                   //VCL
22640: 640 unit uPSI_IdLogEvent;                  //Indy
22641: 641 unit uPSI_Reversi;                     //mX4
```

```
22642: 642 unit uPSI_Geometry;                    //mX4
22643: 643 unit uPSI_IdSMTPServer;                //Indy
22644: 644 unit uPSI_Textures;                    //mX4
22645: 645 unit uPSI_IBX;                         //VCL
22646: 646 unit uPSI_IWDBCommon;                  //VCL
22647: 647 unit uPSI_SortGrid;                    //mX4
22648: 648 unit uPSI_IB;                          //VCL
22649: 649 unit uPSI_IBScript;                    //VCL
22650: 650 unit uPSI_JvCSVBaseControls;           //JCL
22651: 651 unit uPSI_Jvg3DColors;                 //JCL
22652: 652 unit uPSI_JvHLEditor;  //beat          //JCL
22653: 653 unit uPSI_JvShellHook;                 //JCL
22654: 654 unit uPSI_DBCommon2                    //VCL
22655: 655 unit uPSI_JvSHFileOperation;           //JCL
22656: 656 unit uPSI_uFilexport;                  //mX4
22657: 657 unit uPSI_JvDialogs;                   //JCL
22658: 658 unit uPSI_JvDBTreeView;                //JCL
22659: 659 unit uPSI_JvDBUltimGrid;               //JCL
22660: 660 unit uPSI_JvDBQueryParamsForm;         //JCL
22661: 661 unit uPSI_JvExControls;                //JCL
22662: 662 unit uPSI_JvBDEMemTable;               //JCL
22663: 663 unit uPSI_JvCommStatus;                //JCL
22664: 664 unit uPSI_JvMailSlots2;                //JCL
22665: 665 unit uPSI_JvgWinMask;                  //JCL
22666: 666 unit uPSI_StEclpse;                    //SysTools
22667: 667 unit uPSI_StMime;                      //SysTools
22668: 668 unit uPSI_StList;                      //SysTools
22669: 669 unit uPSI_StMerge;                     //SysTools
22670: 670 unit uPSI_StStrS;                      //SysTools
22671: 671 unit uPSI_StTree,                      //SysTools
22672: 672 unit uPSI_StVArr;                      //SysTools
22673: 673 unit uPSI_StRegIni;                    //SysTools
22674: 674 unit uPSI_urkf;                        //DMath
22675: 675 unit uPSI_usvd;                        //DMath
22676: 676 unit uPSI_DepWalkUtils;                //JCL
22677: 677 unit uPSI_OptionsFrm;                  //JCL
22678: 678 unit yuvconverts;                      //mX4
22679: 679 uPSI_JvPropAutoSave;                   //JCL
22680: 680 uPSI_AclAPI;                           //alcinoe
22681: 681 uPSI_AviCap;                           //alcinoe
22682: 682 uPSI_ALAVLBinaryTree;                  //alcinoe
22683: 683 uPSI_ALFcnMisc;                        //alcinoe
22684: 684 uPSI_ALStringList;                     //alcinoe
22685: 685 uPSI_ALQuickSortList;                  //alcinoe
22686: 686 uPSI_ALStaticText;                     //alcinoe
22687: 687 uPSI_ALJSONDoc;                        //alcinoe
22688: 688 uPSI_ALGSMComm;                        //alcinoe
22689: 689 uPSI_ALWindows;                        //alcinoe
22690: 690 uPSI_ALMultiPartFormDataParser;        //alcinoe
22691: 691 uPSI_ALHttpCommon;                     //alcinoe
22692: 692 uPSI_ALWebSpider,                      //alcinoe
22693: 693 uPSI_ALHttpClient;                     //alcinoe
22694: 694 uPSI_ALFcnHTML;                        //alcinoe
22695: 695 uPSI_ALFTPClient;                      //alcinoe
22696: 696 uPSI_ALInternetMessageCommon;          //alcinoe
22697: 697 uPSI_ALWininetHttpClient;              //alcinoe
22698: 698 uPSI_ALWinInetFTPClient;               //alcinoe
22699: 699 uPSI_ALWinHttpWrapper;                 //alcinoe
22700: 700 uPSI_ALWinHttpClient;                  //alcinoe
22701: 701 uPSI_ALFcnWinSock;                     //alcinoe
22702: 702 uPSI_ALFcnSQL;                         //alcinoe
22703: 703 uPSI_ALFcnCGI;                         //alcinoe
22704: 704 uPSI_ALFcnExecute;                     //alcinoe
22705: 705 uPSI_ALFcnFile;                        //alcinoe
22706: 706 uPSI_ALFcnMime;                        //alcinoe
22707: 707 uPSI_ALPhpRunner;                      //alcinoe
22708: 708 uPSI_ALGraphic;                        //alcinoe
22709: 709 uPSI_ALIniFiles;                       //alcinoe
22710: 710 uPSI_ALMemCachedClient;                //alcinoe
22711: 711 unit uPSI_MyGrids;                     //mX4
22712: 712 uPSI_ALMultiPartMixedParser            //alcinoe
22713: 713 uPSI_ALSMTPClient                      //alcinoe
22714: 714 uPSI_ALNNTPClient;                     //alcinoe
22715: 715 uPSI_ALHintBalloon;                    //alcinoe
22716: 716 unit uPSI_ALXmlDoc;                    //alcinoe
22717: 717 unit uPSI_IPCThrd;                     //VCL
22718: 718 unit uPSI_MonForm;                     //VCL
22719: 719 unit uPSI_TeCanvas;                    //Orpheus
22720: 720 unit uPSI_Ovcmisc;                     //Orpheus
22721: 721 unit uPSI_ovcfiler;                    //Orpheus
22722: 722 unit uPSI_ovcstate;                    //Orpheus
22723: 723 unit uPSI_ovccoco;                     //Orpheus
22724: 724 unit uPSI_ovcrvexp;                    //Orpheus
22725: 725 unit uPSI_OvcFormatSettings;           //Orpheus
22726: 726 unit uPSI_OvcUtils;                    //Orpheus
22727: 727 unit uPSI_ovcstore;                    //Orpheus
22728: 728 unit uPSI_ovcstr;                      //Orpheus
22729: 729 unit uPSI_ovcmru;                      //Orpheus
22730: 730 unit uPSI_ovccmd;                      //Orpheus
```

```
22731: 731 unit uPSI_ovctimer;                          //Orpheus
22732: 732 unit uPSI_ovcintl;                           //Orpheus
22733: 733 uPSI_AfCircularBuffer;                       //AsyncFree
22734: 734 uPSI_AfUtils;                                //AsyncFree
22735: 735 uPSI_AfSafeSync;                             //AsyncFree
22736: 736 uPSI_AfComPortCore;                          //AsyncFree
22737: 737 uPSI_AfComPort;                              //AsyncFree
22738: 738 uPSI_AfPortControls;                        //AsyncFree
22739: 739 uPSI_AfDataDispatcher;                      //AsyncFree
22740: 740 uPSI_AfViewers;                             //AsyncFree
22741: 741 uPSI_AfDataTerminal;                        //AsyncFree
22742: 742 uPSI_SimplePortMain;                        //AsyncFree
22743: 743 unit uPSI_ovcclock;                         //Orpheus
22744: 744 unit uPSI_o32intlst;                        //Orpheus
22745: 745 unit uPSI_o32ledlabel;                      //Orpheus
22746: 746 unit uPSI_AlMySqlClient;                    //alcinoe
22747: 747 unit uPSI_ALFBXClient;                      //alcinoe
22748: 748 unit uPSI_ALFcnSQL;                         //alcinoe
22749: 749 unit uPSI_AsyncTimer;                       //mX4
22750: 750 unit uPSI_ApplicationFileIO;                //mX4
22751: 751 unit uPSI_PsAPI;                            //VCLé
22752: 752 uPSI_ovcuser;                               //Orpheus
22753: 753 uPSI_ovcurl;                                //Orpheus
22754: 754 uPSI_ovcvlb;                                //Orpheus
22755: 755 uPSI_ovccolor;                              //Orpheus
22756: 756 uPSI_ALFBXLib,                              //alcinoe
22757: 757 uPSI_ovcmeter;                              //Orpheus
22758: 758 uPSI_ovcpeakm;                              //Orpheus
22759: 759 uPSI_O32BGSty;                              //Orpheus
22760: 760 uPSI_ovcBidi;                               //Orpheus
22761: 761 uPSI_ovctcary;                              //Orpheus
22762: 762 uPSI_DXPUtils;                              //mX4
22763: 763 uPSI_ALMultiPartBaseParser;                 //alcinoe
22764: 764 uPSI_ALMultiPartAlternativeParser;          //alcinoe
22765: 765 uPSI_ALPOP3Client;                          //alcinoe
22766: 766 uPSI_SmallUtils;                            //mX4
22767: 767 uPSI_MakeApp;                               //mX4
22768: 768 uPSI_O32MouseMon;                           //Orpheus
22769: 769 uPSI_OvcCache;                              //Orpheus
22770: 770 uPSI_ovccalc;                               //Orpheus
22771: 771 uPSI_Joystick,                              //OpenGL
22772: 772 uPSI_ScreenSaver;                           //OpenGL
22773: 773 uPSI_XCollection,                           //OpenGL
22774: 774 uPSI_Polynomials,                           //OpenGL
22775: 775 uPSI_PersistentClasses, //9.86              //OpenGL
22776: 776 uPSI_VectorLists;                           //OpenGL
22777: 777 uPSI_XOpenGL,                              //OpenGL
22778: 778 uPSI_MeshUtils;                             //OpenGL
22779: 779 unit uPSI_JclSysUtils;                      //JCL
22780: 780 unit uPSI_JclBorlandTools;                  //JCL
22781: 781 unit JclFileUtils_max;                      //JCL
22782: 782 uPSI_AfDataControls,                        //AsyncFree
22783: 783 uPSI_GLSilhouette;                          //OpenGL
22784: 784 uPSI_JclSysUtils_class;                     //JCL
22785: 785 uPSI_JclFileUtils_class;                    //JCL
22786: 786 uPSI_FileUtil;                              //JCL
22787: 787 uPSI_changefind;                            //mX4
22788: 788 uPSI_cmdIntf;                               //mX4
22789: 789 uPSI_fservice;                              //mX4
22790: 790 uPSI_Keyboard;                              //OpenGL
22791: 791 uPSI_VRMLParser,                            //OpenGL
22792: 792 uPSI_GLFileVRML,                            //OpenGL
22793: 793 uPSI_Octree;                                //OpenGL
22794: 794 uPSI_GLPolyhedron,                          //OpenGL
22795: 795 uPSI_GLCrossPlatform;                       //OpenGL
22796: 796 uPSI_GLParticles;                           //OpenGL
22797: 797 uPSI_GLNavigator;                           //OpenGL
22798: 798 uPSI_GLStarRecord;                          //OpenGL
22799: 799 uPSI_GLTextureCombiners;                    //OpenGL
22800: 800 uPSI_GLCanvas;                              //OpenGL
22801: 801 uPSI_GeometryBB;                            //OpenGL
22802: 802 uPSI_GeometryCoordinates;                   //OpenGL
22803: 803 uPSI_VectorGeometry;                        //OpenGL
22804: 804 uPSI_BumpMapping;                           //OpenGL
22805: 805 uPSI_TGA;                                   //OpenGL
22806: 806 uPSI_GLVectorFileObjects;                   //OpenGL
22807: 807 uPSI_IMM;                                   //VCL
22808: 808 uPSI_CategoryButtons;                       //VCL
22809: 809 uPSI_ButtonGroup;                           //VCL
22810: 810 uPSI_DbExcept;                              //VCL
22811: 811 uPSI_AxCtrls;                               //VCL
22812: 812 uPSI_GL_actorUnit1;                         //OpenGL
22813: 813 uPSI_StdVCL;                                //VCL
22814: 814 unit CurvesAndSurfaces;                     //OpenGL
22815: 815 uPSI_DataAwareMain;                         //AsyncFree
22816: 816 uPSI_TabNotBk;                              //VCL
22817: 817 uPSI_udwsfiler;                             //mX4
22818: 818 uPSI_synaip;                                //Synapse!
22819: 819 uPSI_synacode;                              //Synapse
```

```
22820: 820 uPSI_synachar;                    //Synapse
22821: 821 uPSI_synamisc;                    //Synapse
22822: 822 uPSI_synaser;                     //Synapse
22823: 823 uPSI_synaicnv;                    //Synapse
22824: 824 uPSI_tlntsend;                    //Synapse
22825: 825 uPSI_pingsend;                    //Synapse
22826: 826 uPSI_blcksock;                    //Synapse
22827: 827 uPSI_asn1util;                    //Synapse
22828: 828 uPSI_dnssend;                     //Synapse
22829: 829 uPSI_clamsend;                    //Synapse
22830: 830 uPSI_ldapsend;                    //Synapse
22831: 831 uPSI_mimemess;                    //Synapse
22832: 832 uPSI_slogsend;                    //Synapse
22833: 833 uPSI_mimepart;                    //Synapse
22834: 834 uPSI_mimeinln;                    //Synapse
22835: 835 uPSI_ftpsend,                     //Synapse
22836: 836 uPSI_ftptsend;                    //Synapse
22837: 837 uPSI_httpsend;                    //Synapse
22838: 838 uPSI_sntpsend;                    //Synapse
22839: 839 uPSI_smtpsend;                    //Synapse
22840: 840 uPSI_snmpsend;                    //Synapse
22841: 841 uPSI_imapsend;                    //Synapse
22842: 842 uPSI_pop3send;                    //Synapse
22843: 843 uPSI_nntpsend;                    //Synapse
22844: 844 uPSI_ssl_cryptlib;                //Synapse
22845: 845 uPSI_ssl_openssl;                 //Synapse
22846: 846 uPSI_synhttp_daemon;              //Synapse
22847: 847 uPSI_NetWork;                     //mX4
22848: 848 uPSI_PingThread;                  //Synapse
22849: 849 uPSI_JvThreadTimer;               //JCL
22850: 850 unit uPSI_wwSystem;               //InfoPower
22851: 851 unit uPSI_IdComponent;            //Indy
22852: 852 unit uPSI_IdIOHandlerThrottle;    //Indy
22853: 853 unit uPSI_Themes;                 //VCL
22854: 854 unit uPSI_StdStyleActnCtrls;      //VCL
22855: 855 unit uPSI_UDDIHelper;             //VCL
22856: 856 unit uPSI_IdIMAP4Server;          //Indy
22857: 857 uPSI_VariantSymbolTable,          //VCL //3.9.9.92
22858: 858 uPSI_udf_glob,                    //mX4
22859: 859 uPSI_TabGrid,                     //VCL
22860: 860 uPSI_JsDBTreeView,               //mX4
22861: 861 uPSI_JsSendMail,                 //mX4
22862: 862 uPSI_dbTvRecordList,             //mX4
22863: 863 uPSI_TreeVwEx,                   //mX4
22864: 864 uPSI_ECDataLink,                 //mX4
22865: 865 uPSI_dbTree,                     //mX4
22866: 866 uPSI_dbTreeCBox,                 //mX4
22867: 867 unit uPSI_Debug; //TfrmDebug     //mX4
22868: 868 uPSI_TimeFncs;                   //mX4
22869: 869 uPSI_FileIntf,                   //VCL
22870: 870 uPSI_SockTransport,              //RTL
22871: 871 unit uPSI_WinInet;               //RTL
22872: 872 unit uPSI_Wwstr;                 //mX4
22873: 873 uPSI_DBLookup,                   //VCL
22874: 874 uPSI_Hotspot,                    //mX4
22875: 875 uPSI_HList; //History List       //mX4
22876: 876 unit uPSI_DrTable;               //VCL
22877: 877 uPSI_TConnect,                   //VCL
22878: 878 uPSI_DataBkr,                    //VCL
22879: 879 uPSI_HTTPIntr;                   //VCL
22880: 880 unit uPSI_Mathbox;               //mX4
22881: 881 uPSI_cyIndy,                     //cY
22882: 882 uPSI_cySysUtils,                 //cY
22883: 883 uPSI_cyWinUtils,                 //cY
22884: 884 uPSI_cyStrUtils,                 //cY
22885: 885 uPSI_cyObjUtils,                 //cY
22886: 886 uPSI_cyDateUtils,                //cY
22887: 887 uPSI_cyBDE,                      //cY
22888: 888 uPSI_cyClasses,                  //cY
22889: 889 uPSI_cyGraphics,  //3.9.9.94_2   //cY
22890: 890 unit uPSI_cyTypes;               //cY
22891: 891 uPSI_JvDateTimePicker,           //JCL
22892: 892 uPSI_JvCreateProcess,            //JCL
22893: 893 uPSI_JvEasterEgg,                //JCL
22894: 894 uPSI_WinSvc,  //3.9.9.94_3       //VCL
22895: 895 uPSI_SvcMgr                      //VCL
22896: 896 unit uPSI_JvPickDate;            //JCL
22897: 897 unit uPSI_JvNotify;              //JCL
22898: 898 uPSI_JvStrHlder                  //JCL
22899: 899 unit uPSI_JclNTFS2;              //JCL
22900: 900 uPSI_Jcl8087 //math coprocessor  //JCL
22901: 901 uPSI_JvAddPrinter                //JCL
22902: 902 uPSI_JvCabFile                   //JCL
22903: 903 uPSI_JvDataEmbedded;             //JCL
22904: 904 unit uPSI_U_HexView;             //mX4
22905: 905 uPSI_UWavein4,                   //mX4
22906: 906 uPSI_AMixer,                     //mX4
22907: 907 uPSI_JvaScrollText,              //mX4
22908: 908 uPSI_JvArrow,                    //mX4
```

```
22909: 909 unit uPSI_UrlMon;                          //mX4
22910: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'   //mX4
22911: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
22912: 912 unit uPSI_DFFUtils;                        //DFF
22913: 913 unit uPSI_MathsLib;                        //DFF
22914: 914 uPSI_UIntList;                             //DFF
22915: 915 uPSI_UGetParens;                           //DFF
22916: 916 unit uPSI_UGeometry;                       //DFF
22917: 917 unit uPSI_UAstronomy;                      //DFF
22918: 918 unit uPSI_UCardComponentV2;                //DFF
22919: 919 unit uPSI_UTGraphSearch;                   //DFF
22920: 920 unit uPSI_UParser10;                       //DFF
22921: 921 unit uPSI_cyIEUtils;                       //cY
22922: 922 unit uPSI_UcomboV2;                        //DFF
22923: 923 uPSI_cyBaseComm,                           //cY
22924: 924 uPSI_cyAppInstances,                       //cY
22925: 925 uPSI_cyAttract,                           //cY
22926: 926 uPSI_cyDERUtils                           //cY
22927: 927 unit uPSI_cyDocER;                        //cY
22928: 928 unit uPSI_ODBC;                           //mX
22929: 929 unit uPSI_AssocExec;                      //mX
22930: 930 uPSI_cyBaseCommRoomConnector,             //cY
22931: 931 uPSI_cyCommRoomConnector,                 //cY
22932: 932 uPSI_cyCommunicate,                       //cY
22933: 933 uPSI_cyImage;                            //cY
22934: 934 uPSI_cyBaseContainer                     //cY
22935: 935 uPSI_cyModalContainer,                    //cY
22936: 936 uPSI_cyFlyingContainer;                   //cY
22937: 937 uPSI_RegStr,                             //VCL
22938: 938 uPSI_HtmlHelpViewer;                     //VCL
22939: 939 unit uPSI_cyIniForm                      //cY
22940: 940 unit uPSI_cyVirtualGrid;                 //cY
22941: 941 uPSI_Profiler,                           //DA
22942: 942 uPSI_BackgroundWorker,                   //DA
22943: 943 uPSI_WavePlay,                           //DA
22944: 944 uPSI_WaveTimer,                          //DA
22945: 945 uPSI_WaveUtils;                         //DA
22946: 946 uPSI_NamedPipes,                         //TB
22947: 947 uPSI_NamedPipeServer,                    //TB
22948: 948 unit uPSI_process,                       //TB
22949: 949 unit uPSI_DPUtils;                       //TB
22950: 950 unit uPSI_CommonTools;                   //TB
22951: 951 uPSI_DataSendToWeb,                      //TB
22952: 952 uPSI_StarCalc,                          //TB
22953: 953 uPSI_D2_XPVistaHelperU                   //TB
22954: 954 unit uPSI_NetTools                      //TB
22955: 955 unit uPSI_Pipes                         //TB
22956: 956 uPSI_ProcessUnit,                       //mX
22957: 957 uPSI_adGSM,                             //mX
22958: 958 unit uPSI_BetterADODataSet;             //mX
22959: 959 unit uPSI_AdSelCom; //FTT               //mX
22960: 960 unit unit uPSI_dwsXPlatform;            //DWS
22961: 961 uPSI_AdSocket;                          //mX Turbo Power
22962: 962 uPSI_AdPacket;                          //mX
22963: 963 uPSI_AdPort;                            //mX
22964: 964 uPSI_PathFunc;                          //Inno
22965: 965 uPSI_CmnFunc;                           //Inno
22966: 966 uPSI_CmnFunc2; //Inno Setup             //Inno
22967: 967 unit uPSI_BitmapImage;                  //mX4
22968: 968 unit uPSI_ImageGrabber;                 //mX4
22969: 969 uPSI_SecurityFunc,                      //Inno
22970: 970 uPSI_RedirFunc,                         //Inno
22971: 971 uPSI_FIFO, (MemoryStream)               //mX4
22972: 972 uPSI_Int64Em,                           //Inno
22973: 973 unit uPSI_InstFunc;                     //Inno
22974: 974 unit uPSI_LibFusion;                    //Inno
22975: 975 uPSI_SimpleExpression;                  //Inno
22976: 976 uPSI_unitResourceDetails,               //XN
22977: 977 uPSI_unitResFile,                       //XN
22978: 978 unit uPSI_simpleComport;               //mX4
22979: 979 unit uPSI_AfViewershelpers;            //Async
22980: 980 unit uPSI_Console;                      //mX4
22981: 981 unit uPSI_AnalogMeter;                  //TB
22982: 982 unit uPSI_XPrinter,                     //TB
22983: 983 unit uPSI_IniFiles;                     //VCL
22984: 984 unit uPSI_lazIniFiles;                  //FP
22985: 985 uPSI_testutils;                        //FP
22986: 986 uPSI_ToolsUnit; (DBTests)              //FP
22987: 987 uPSI_fpcunit                           //FP
22988: 988 uPSI_testdecorator;                    //FP
22989: 989 unit uPSI_fpcunittests;                //FP
22990: 990 unit uPSI_cTCPBuffer;                   //Fundamentals 4
22991: 991 unit uPSI_Glut,                        //Open GL
22992: 992 uPSI_LEDBitmaps,                       //mX4
22993: 993 uPSI_FileClass,                        //Inno
22994: 994 uPSI_FileUtilsClass,                   //mX4
22995: 995 uPSI_ComPortInterface; //Kit           //mX4
22996: 996 unit uPSI_SwitchLed;                   //mX4
22997: 997 unit uPSI_cyDmmCanvas;                 //cY
```

```
22998: 998 uPSI_uColorFunctions;                    //DFF
22999: 999 uPSI_uSettings;                          //DFF
23000: 1000 uPSI_cyDebug.pas                        //cY
23001: 1001 uPSI_cyColorMatrix;                     //cY
23002: 1002 unit uPSI_cyCopyFiles;                  //cY
23003: 1003 unit uPSI_cySearchFiles;                //cY
23004: 1004 unit uPSI_cyBaseMeasure;                //cY
23005: 1005 unit uPSI_PJIStreams;                   //DD
23006: 1006 unit uPSI_cyRunTimeResize;              //cY
23007: 1007 unit uPSI_jcontrolutils;                //cY
23008: 1008 unit uPSI_kcMapViewer; (+GEONames)      //kc
23009: 1009 unit uPSI_kcMapViewerDESynapse;         //kc
23010: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
23011: 1011 unit uPSI_LedNumber;                    //TurboPower
23012: 1012 unit uPSI_StStrL;                       //SysTools
23013: 1013 unit uPSI_indGnouMeter;                 //LAZ
23014: 1014 unit uPSI_Sensors;                      //LAZ
23015: 1015 unit uPSI_pwmain;  //cgi of powtils     //Pow
23016: 1016 unit uPSI_HTMLUtil;                     //Pow
23017: 1017 unit uPSI_synwrap1; //httpsend          //Pow
23018: 1018 unit StreamWrap1                        //Pow
23019: 1019 unit uPSI_pwmain;                       //Pow
23020: 1020 unit pwtypes                           //Pow
23021: 1021 uPSI_W32VersionInfo                    //LAZ
23022: 1022 unit uPSI_IpAnim;                      //LAZ
23023: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
23024: 1024 unit uPSI_LrtPoTools;                  //LAZ
23025: 1025 unit uPSI_Laz_DOM;                     //LAZ
23026: 1026 unit uPSI_hhAvComp;                    //LAZ
23027: 1027 unit uPSI_GPS2;                        //mX4
23028: 1028 unit uPSI_GPS;                         //mX4
23029: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo//mX4
23030: 1030 unit uPSI_NMEA;     // GPS              //mX4
23031: 1031 unit uPSI_ScreenThreeDLab;             //mX4
23032: 1032 unit uPSI_Spin;                        //VCL
23033: 1033 unit uPSI_DynaZip;                     //mX4
23034: 1034 unit uPSI_clockExpert;                 //TB
23035: 1035 unit debugLn                           //mX4
23036: 1036 uPSI_SortUtils;                       //Jcl
23037: 1037 uPSI_BitmapConversion;                 //Jcl
23038: 1038 unit uPSI_JclTD32;                     //Jcl
23039: 1039 unit uPSI_ZDbcUtils;                   //Zeos
23040: 1040 unit uPSI_ZScriptParser;               //Zeos
23041: 1041 uPSI_JvIni,                           //JCL
23042: 1042 uPSI_JvFtpGrabber;                    //JCL
23043: 1043 unit uPSI_NeuralNetwork;               //OCL
23044: 1044 unit uPSI_StExpr;                      //SysTools
23045: 1045 unit uPSI_GR32_Geometry;              //GR32
23046: 1046 unit uPSI_GR32_Containers;            //GR32
23047: 1047 unit uPSI_GR32_Backends_VCL,          //GR32
23048: 1048 unit uPSI_StSaturn;//Venus+Mercury+Mars++ //SysTools
23049: 1049 unit uPSI_JclParseUses;               //JCL
23050: 1050 unit uPSI_JvFinalize;                 //JCL
23051: 1051 unit uPSI_panUnit1;                   //GLScene
23052: 1052 unit uPSI_DD83u1;  //Arduino Tester   //mX4
23053: 1053 unit uPSI_BigIni                      //Hinzen
23054: 1054 unit uPSI_ShellCtrls;                 //VCL
23055: 1055 unit uPSI_fmath;                      //FMath
23056: 1056 unit uPSI_fComp;                      //FMath
23057: 1057 unit uPSI_HighResTimer;               //Lauer
23058: 1058 unit uconvMain; (Unit Converter)      //PS
23059: 1059 unit uPSI_uconvMain;                  //PS
23060: 1060 unit uPSI_ParserUtils;                //PS
23061: 1061 unit uPSI_uPSUtils;                   //PS
23062:
23063:
23064: ////////////////////////////////////////////////////////////////////////////
23065: //Form Template Library FTL
23066: ////////////////////////////////////////////////////////////////////////////
23067:
23068: 36 FTL For Form Building out of the Script, eg. 399_form_templates.txt
23069: 045 unit uPSI_VListView                      TFormListView;
23070: 263 unit uPSI_JvProfiler32;                  TProfReport
23071: 270 unit uPSI_ImgList;                       TCustomImageList
23072: 278 unit uPSI_JvImageWindow;                 TJvImageWindow
23073: 317 unit uPSI_JvParserForm;                  TJvHTMLParserForm
23074: 497 unit uPSI_DebugBox;                      TDebugBox
23075: 513 unit uPSI_ImageWin;                      TImageForm, TImageForm2
23076: 514 unit uPSI_CustomDrawTreeView;           TCustomDrawForm
23077: 515 unit uPSI_GraphWin;                      TGraphWinForm
23078: 516 unit uPSI_actionMain;                    TActionForm
23079: 518 unit uPSI_CtlPanel;                     TAppletApplication
23080: 529 unit uPSI_MDIEdit;                      TEditForm
23081: 535 unit uPSI_CoolMain; {browser}           TWebMainForm
23082: 538 unit uPSI_frmExportMain;                TSynexportForm
23083: 585 unit uPSI_usniffer; {//PortScanForm}    TSniffForm
23084: 600 unit uPSI_ThreadForm;                   TThreadSortForm;
23085: 618 unit uPSI_delphi_arduino_Unit1;         TLEDForm
23086: 620 unit uPSI_fplotMain;                    TfplotForm1
```

```
23087: 660 unit uPSI_JvDBQueryParamsForm;            TJvQueryParamsDialog
23088: 677 unit uPSI_OptionsFrm;                     TfrmOptions;
23089: 718 unit uPSI_MonForm;                        TMonitorForm
23090: 742 unit uPSI_SimplePortMain;                 TPortForm1
23091: 770 unit uPSI_ovccalc;                        TOvcCalculator  //widget
23092: 810 unit uPSI_DbExcept;                       TDbEngineErrorDlg
23093: 812 unit uPSI_GL_actorUnit1;                  TglActorForm1  //OpenGL Robot
23094: 846 unit uPSI_synhttp_daemon;                 TTCPHttpDaemon, TTCPHttpThrd, TPingThread
23095: 867 unit uPSI_Debug;                          TfrmDebug
23096: 904 unit uPSI_U_HexView;                      THexForm2
23097: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)  TOscfrmMain
23098: 959 unit uPSI_AdSelCom;                       TComSelectForm
23099: 1029 unit uPSI_GPSUDemo;                      TFDemo
23100: 1031 unit uPSI_ScreenThreeDLab;               TFormLab3D
23101: 1051 unit uPSI_panUnit1;                      TPanForm1 //GLScene
23102: 1052 unit uPSI_DD83u1;  {Arduino Tester Frm}  TDD83f1
23103: 1059 unit uPSI_uconvMain;                     TfconvMain //PS
23104:
23105: ex.:with TEditForm.create(self) do begin
23106:       caption:= 'Template Form Tester';
23107:       FormStyle:= fsStayOnTop;
23108:       with editor do begin
23109:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
23110:         SelStart:= 0;
23111:         Modified:= False;
23112:       end;
23113:     end;
23114:   with TWebMainForm.create(self) do begin
23115:     URLs.Text:= 'http://www.kleiner.ch';
23116:     URLsClick(self); Show;
23117:   end;
23118:   with TSynexportForm.create(self) do begin
23119:     Caption:= 'Synexport HTML RTF tester';
23120:     Show;
23121:   end;
23122:   with TThreadSortForm.create(self) do begin
23123:      showmodal; free;
23124:   end;
23125:   with TCustomDrawForm.create(self) do begin
23126:        width:=820; height:=820;
23127:        image1.height:= 600; //add properties
23128:        image1.picture.bitmap:= image2.picture.bitmap;
23129:        //SelectionBackground1Click(self) CustomDraw1Click(self);
23130:        Background1.click;
23131:        bitmap1.click; Tile1.click;
23132:        Showmodal;
23133:        Free;
23134:      end;
23135:    with TfplotForm1.Create(self) do begin
23136:      BtnPlotClick(self);
23137:      Showmodal; Free;
23138:    end;
23139:   with TOvcCalculator.create(self) do begin
23140:     parent:= aForm;  //free;
23141:     setbounds(550,510,200,150);
23142:     displaystr:= 'maXcalc';
23143:   end;
23144:   with THexForm2.Create(self) do begin
23145:      ShowModal;
23146:      Free;
23147:   end;
23148:
23149: function CheckBox: string;
23150: var idHTTP: TIDHTTP;
23151:  begin
23152:    result:= 'version not found';
23153:    if IsInternet then begin
23154:      idHTTP:= TIdHTTP.Create(NIL);
23155:      try
23156:        result:= idHTTP.Get(MXVERSIONFILE2);
23157:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
23158:        if result = MBVER2 then begin
23159:          //Speak(' A new Version '+vstr+' of max box is available! ');
23160:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
23161:        end;  //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
23162:      finally
23163:        idHTTP.Free
23164:    end;
23165:  end;
23166: end;
23167:
23168: //Runtimer Functions Edition
23169: function ApWinExecAndWait32(FileName:PChar; CommandLine:PChar; Visibility:Integer):Integer;
23170: function KillTask(ExeFileName: string): Integer;
23171: procedure KillProcess(hWindowHandle: HWND);
23172: function FindWindowByTitle(WindowTitle: string): Hwnd;
23173: function IntToFloat(i: Integer): double;
23174: function AddThousandSeparator(S: string; myChr: Char): string;
23175: function mciSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
```

```
23176: procedure FormAnimation(Sender: TObject; adelay: integer);
23177: procedure LoadResourceFile2(aFile:string; ms:TMemoryStream);
23178: function putBinResTo(binresname: pchar; newpath: string): boolean;
23179: procedure ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:string);
23180: function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string; MouseX,MouseY:Integer;var
       HyperLink:string):Bool
23181: Function GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD) : DWORD;');
23182: Function GetWindowTask( hWnd : HWND) : THandle');
23183: Function LoadBitmap( hInstance : HINST; lpBitmapName : PChar) : HBITMAP');
23184: Function GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL');
23185: function WinExecAndWait32(FileName: string; Visibility: Integer): Longword;
23186: Function MakeHash( const s : TbtString) : Longint');
23187: Function GetUsedUnitList( list : Tstringlist) : string');
23188:
23189:   command1:= 'play "'+songpath+'maxbox.wav"';
23190:   command2:= 'play "'+songpath+'moon.wav"';
23191:   SendMCICommand('open waveaudio shareable');   //parallels
23192:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav"');
23193:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav"');
23194:   SendMCICommand('close waveaudio');
23195:
23196:
23197: ////////////////////////////////////////////////////////////////////////////
23198: All maXbox Tutorials Table of Content 2014
23199: ////////////////////////////////////////////////////////////////////////////
23200:     Tutorial 00 Function-Coding  (Blix the Programmer)
23201:     Tutorial 01 Procedural-Coding
23202:     Tutorial 02 OO-Programming
23203:     Tutorial 03 Modular Coding
23204:     Tutorial 04 UML Use Case Coding
23205:     Tutorial 05 Internet Coding
23206:     Tutorial 06 Network Coding
23207:     Tutorial 07 Game Graphics Coding
23208:     Tutorial 08 Operating System Coding
23209:     Tutorial 09 Database Coding
23210:     Tutorial 10 Statistic Coding
23211:     Tutorial 11 Forms Coding
23212:     Tutorial 12 SQL DB Coding
23213:     Tutorial 13 Crypto Coding
23214:     Tutorial 14 Parallel Coding
23215:     Tutorial 15 Serial RS232 Coding
23216:     Tutorial 16 Event Driven Coding
23217:     Tutorial 17 Web Server Coding
23218:     Tutorial 18 Arduino System Coding
23219:     Tutorial 18_3 RGB LED System Coding
23220:     Tutorial 19 WinCOM /Arduino Coding
23221:     Tutorial 20 Regular Expressions RegEx
23222:     Tutorial 21 Android Coding (coming 2013)
23223:     Tutorial 22 Services Programming
23224:     Tutorial 23 Real Time Systems
23225:     Tutorial 24 Clean Code
23226:     Tutorial 25 maXbox Configuration I+II
23227:     Tutorial 26 Socket Programming with TCP
23228:     Tutorial 27 XML & TreeView
23229:     Tutorial 28 DLL Coding (available)
23230:     Tutorial 29 UML Scripting (2014)
23231:     Tutorial 30 Web of Things (2014)
23232:     Tutorial 31 Closures (2014)
23233:     Tutorial 32 SQL Firebird (coming 2014)
23234:     Tutorial 33 Oscilloscope (coming 2015)
23235:     Tutorial 34 GPS Navigation (2014)
23236:     Tutorial 35 Web Box (available)
23237:     Tutorial 36 Unit Testing (coming 2015)
23238:     Tutorial 37 API Coding (coming 2015)
23239:     Tutorial 38 3D Coding (coming 2015)
23240:     Tutorial 39 GEO Map Coding (available)
23241:     Tutorial 39_1 GEO Map Layers Coding (available)
23242:     Tutorial 40 REST Coding (coming 2015)
23243:
23244:
23245: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
23246: using Docu for this file is maxbox_functions_all.pdf
23247: PEP - Pascal Education Program  Low Lib Lab ShellHell in UDEMY
23248:
23249: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
23250: http://stackoverflow.com/tags/pascalscript/hot
23251: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
23252: http://sourceforge.net/projects/maXbox   #locs:51620
23253: http://sourceforge.net/apps/mediawiki/maXbox
23254: http://www.blaisepascal.eu/
23255: https://github.com/maxkleiner/maXbox3.git
23256: http://www.heise.de/download/maxbox-1176464.html
23257: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
23258: https://www.facebook.com/pages/Programming-maXbox/166844836691703
23259: http://www.softwareschule.ch/arduino_training.pdf
23260: http://www.delphiarea.com
23261: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
23262: http://entwickler-konferenz.de/2014/speakers/max-kleiner
23263: http://www.heise.de/download/maxbox-1176464.html
```

https://www.udemy.com/learn-coding-from-the-scratch
http://www.slideshare.net/maxkleiner1/codesign-2015

All **maXbox** Examples List
https://github.com/maxkleiner/maXbox3/releases

000_pas_baseconvert.txt	282_fadengraphik.txt
000_pas_baseconvert.txt_encrypt	283_SQL_API_messagetimeout.txt
000_pas_baseconvert.txt_decrypt	284_SysTools4.txt
001_1_pas_functest - Kopie.txt	285_MineForm_GR32.TXT
001_1_pas_functest.txt	285_MineForm_GR32main.TXT
001_1_pas_functest2.txt	285_MineForm_GR32mainsolution.TXT
001_1_pas_functest_clx2.txt	285_MineForm_propas.TXT
001_1_pas_functest_clx2_2.txt	285_MineForm_propas2.TXT
001_1_pas_functest_openarray.txt	285_minesweeper2.TXT
001_pas_lottogen.txt	285_Patterns_process.txt
001_pas_lottogen_template.txt	286_colormixer_jpeg_charcounter.txt
001_pas_lottogen.txtcopy	286_colormixer_jpeg_charcounter2.txt
002_pas_russianroulette.txt	287_eventhandling.txt
002_pas_russianroulette.txtcopy	287_eventhandling2.txt
002_pas_russianroulette.txtcopy_decrypt	287_eventhandling2_negpower.txt
002_pas_russianroulette.txtcopy_encrypt	288_bitblt.txt
003_pas_motion.txt	288_bitblt_resize.txt
003_pas_motion.txtcopy	289_regression.txt
004_pas_search.txt	289_regression2.txt
004_pas_search_replace.txt	290_bestofbox.txt
004_search_replace_allfunctionlist.txt	290_bestofbox2.txt
005_pas_oodesign.txt	290_bestofbox3.txt
005_pas_shelllink.txt	291_3sort_visual_thread.txt
006_pas_oobatch.txt	292_refactoring2.txt
007_pas_streamcopy.txt	293_bold_utils.txt
008_EINMALEINS_FUNC.TXT	293_ib_utils.txt
008_explanation.txt	293_ib_utils_timetest.txt
008_pas_verwechselt.txt	294_maxcalc_demo.txt
008_pas_verwechselt_ibz_bern_func.txt	294_maxcalc_demo2.txt
008_stack_ibz.TXT	295_easter_calendar.txt
009_pas_umlrunner.txt	295_easter_calendar2.txt
009_pas_umlrunner_all.txt	295_easter_combobox.txt
009_pas_umlrunner_componenttest.txt	297_atomimage.txt
009_pas_umlrunner_solution.txt	297_atomimage2.txt
009_pas_umlrunner_solution_2step.txt	297_atomimage3.txt
010_pas_oodesign_solution.txt	297_atomimage4.txt
011_pas_puzzlepas_defect.txt	297_maxonmotor.TXT
012_pas_umlrunner_solution.txt	297_maxon_atomimage9.txt
012_pas_umlrunner_solution2.txt	298_bitblt_animation.txt
013_pas_linenumber.txt	298_bitblt_animation2.txt
014_pas_primetest.txt	298_bitblt_animation3.txt
014_pas_primetest_first.txt	298_bitblt_animation4.txt
014_pas_primetest_sync.txt	298_bitblt_animation4_screensaver.txt
015_pas_designbycontract.txt	298_bitblt_animation5_screensaver.txt
015_pas_designbycontract_solution.txt	299_animation.txt
016_pas_searchrec.txt	299_animationmotor_arduino.txt
017_chartgen.txt	299_animation_formprototype.txt
018_data_simulator.txt	299_realtimeclock_arduino.txt
019_dez_to_bin.txt	299_realtimeclock_arduino2.txt
019_dez_to_bin_grenzwert_ibz.txt	300_treeview.txt
020_proc_feedback.txt	300_treeview_test.txt
021_pas_symkey.txt	300_treeview_test2.txt
021_pas_symkey_solution.txt	300_treeview_test3.txt
022_pas_filestreams.txt	301_LED_Arduino3.txt
023_pas_find_searchrec.txt	301_led_arduino3_simple.txt
023_pas_pathfind.txt	301_led_arduino3_simplecode.txt
024_pas_TFileStream_records.txt	301_log_arduino.txt
025_prime_direct.txt	301_log_arduino2.txt
026_pas_memorystream.txt	301_SQL_DBfirebird3.txt
027_pas_shellexecute_beta.txt	301_SQL_DBfirebird4.txt
027_pas_shellexecute_solution.txt	302_LCLActivity_java.txt
028_pas_dataset.txt	302_LED_DataLogger.txt
029_pas_assignfile.txt	303_Android_LCLActivity_java.txt
029_pas_assignfile_dragndropexe.txt	303_webserver.txt
030_palindrome_2.txt	303_webserver2.txt
030_palindrome_tester.txt	303_webserver_alldocs2.txt
030_pas_recursion.txt	303_webserver_alldocs2_tester.txt
030_pas_recursion2.txt	303_webserver_minimal.txt
031_pas_hashcode.txt	303_webserver_simple.txt
032_pas_crc_const.txt	304_st_system.txt
033_pas_cipher.txt	305_indy_elizahttpserver.TXT
033_pas_cipher_def.txt	305_indy_elizahttpserver2.TXT
033_pas_cipher_file_2_solution.txt	305_indy_elizahttpserver3.TXT
034_pas_soundbox.txt	305_indy_elizahttpserver4file.TXT
035_pas_crcscript.txt	305_webserver_arduino.txt
035_pas_CRCscript_modbus.txt	305_webserver_arduino2.txt
036_pas_includetest.txt	305_webserver_arduino3.txt
036_pas_includetest_basta.txt	305_webserver_arduino3ibz.txt
037_pas_define_demo32.txt	305_webserver_arduino3ibz_rgb_led.txt
038_pas_box_demonstrator.txt	305_webserver_arduino3test.txt
039_pas_dllcall.txt	306_SPS_http_command.txt

23353: 040_paspointer.txt
23354: 040_paspointer_old.txt
23355: 041_pasplotter.txt
23356: 041_pasplotter_plus.txt
23357: 042_pas_kgv_ggt.txt
23358: 043_pas_proceduretype.txt
23359: 044_pas_14queens_solwith14.txt
23360: 044_pas_8queens.txt
23361: 044_pas_8queens_sol2.txt
23362: 044_pas_8queens_solutions.txt
23363: 044_queens_performer.txt
23364: 044_queens_performer2.txt
23365: 044_queens_performer2tester.txt
23366: 045_pas_listhandling.txt
23367: 046_pas_records.txt
23368: 047_pas_modula10.txt
23369: 048_pas_romans.txt
23370: 049_pas_ifdemo.txt
23371: 049_pas_ifdemo_BROKER.txt
23372: 050_pas_primetest2.txt
23373: 050_pas_primetester_thieves.txt
23374: 050_program_starter.txt
23375: 050_program_starter_performance.txt
23376: 051_pas_findtext_solution.txt
23377: 052_pas_text_as_stream.txt
23378: 052_pas_text_as_stream_include.txt
23379: 053_pas_singleton.txt
23380: 054_pas_speakpassword.txt
23381: 054_pas_speakpassword2.txt
23382: 054_pas_speakpassword_searchtest.txt
23383: 055_pas_factorylist.txt
23384: 056_pas_demeter.txt
23385: 057_pas_dirfinder.txt
23386: 058_pas_filefinder.txt
23387: 058_pas_filefinder_pdf.txt
23388: 058_pas_filefinder_screview.txt
23389: 058_pas_filefinder_screview2.txt
23390: 058_pas_filefinder_screview3.txt
23391: 059_pas_timertest.txt
23392: 059_pas_timertest_2.txt
23393: 059_pas_timertest_time_solution.txt
23394: 059_timerobject_starter2.txt
23395: 059_timerobject_starter2_ibz2_async.txt
23396: 059_timerobject_starter2_uml.txt
23397: 059_timerobject_starter2_uml_main.txt
23398: 059_timerobject_starter4_ibz.txt
23399: 060_pas_datefind.txt
23400: 060_pas_datefind_exceptions2.txt
23401: 060_pas_datefind_exceptions_CHECKTEST.txt
23402: 060_pas_datefind_fulltext.txt
23403: 060_pas_datefind_plus.txt
23404: 060_pas_datefind_plus_mydate.txt
23405: 061_pas_randomwalk.txt
23406: 061_pas_randomwalk_plus.txt
23407: 062_paskorrelation.txt
23408: 063_pas_calculateform.txt
23409: 063_pas_calculateform_2list.txt
23410: 064_pas_timetest.txt
23411: 065_pas_bitcounter.txt
23412: 066_pas_eliza.txt
23413: 066_pas_eliza_include_sol.txt
23414: 067_pas_morse.txt
23415: 068_pas_piezo_sound.txt
23416: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
23417: 069_my_LEDBOX.TXT
23418: 069_pas_ledmatrix.txt
23419: 069_pas_LEDMATRIX_Alphabet.txt
23420: 069_pas_LEDMATRIX_Alphabet_run.txt
23421: 069_pas_LEDMATRIX_Alphabet_tester.txt
23422: 069_PAS_LEDMATRIX_COLOR.TXT
23423: 069_pas_ledmatrix_fixedit.txt
23424: 069_pas_LEDMATRIX_soundbox.txt
23425: 069_pas_LEDMATRIX_soundbox2.txt
23426: 069_Richter_MATRIX.TXT
23427: 070_pas_functionplot.txt
23428: 070_pas_functionplotter2.txt
23429: 070_pas_functionplotter2_mx4.txt
23430: 070_pas_functionplotter2_tester.txt
23431: 070_pas_functionplotter3.txt
23432: 070_pas_functionplotter4.txt
23433: 070_pas_functionplotter_digital.txt
23434: 070_pas_functionplotter_elliptic.txt
23435: 070_pas_function_helmholtz.txt
23436: 070_pas_textcheck_experimental.txt
23437: 071_pas_graphics.txt
23438: 071_pas_graphics_drawsym.txt
23439: 071_pas_graphics_drawsym_save.txt
23440: 071_pas_graphics_random.txt
23441: 072_pas_fractals.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
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

23442: 072_pas_fractals_2.txt
23443: 072_pas_fractals_blackhole.txt
23444: 072_pas_fractals_perfomance.txt
23445: 072_pas_fractals_perfomance_new.txt
23446: 072_pas_fractals_perfomance_sharp.txt
23447: 072_pas_fractals_performance.txt
23448: 072_pas_fractals_performance_mX4.txt
23449: 073_pas_forms.txt
23450: 074_pas_chartgenerator.txt
23451: 074_pas_chartgenerator_solution.txt
23452: 074_pas_chartgenerator_solution_back.txt
23453: 074_pas_charts.txt
23454: 075_bitmap_Artwork2.txt
23455: 075_pas_bitmappuzzle.txt
23456: 075_pas_bitmappuzzle24.prod.txt
23457: 075_pas_bitmappuzzle2_prod.txt
23458: 075_pas_bitmappuzzle3.txt
23459: 075_pas_bitmapsolve.txt
23460: 075_pas_bitmap_Artwork.txt
23461: 075_pas_puzzlepas_solution.txt
23462: 076_pas_3dcube.txt
23463: 076_pas_circle.txt
23464: 077_pas_mmshow.txt
23465: 078_pas_pi.txt
23466: 079_pas_3dcube_animation.txt
23467: 079_pas_3dcube_animation4.txt
23468: 079_pas_3dcube_plus.txt
23469: 080_pas_hanoi.txt
23470: 080_pas_hanoi2.txt
23471: 080_pas_hanoi2_file.txt
23472: 080_pas_hanoi2_sol.txt
23473: 080_pas_hanoi2_tester.txt
23474: 080_pas_hanoi2_tester_fast.txt
23475: 080_pas_hanoi3.txt
23476: 081_pas_chartist2.txt
23477: 082_pas_biorythmus.txt
23478: 082_pas_biorythmus_solution.txt
23479: 082_pas_biorythmus_solution_3.txt
23480: 082_pas_biorythmus_test.txt
23481: 083_pas_GITARRE.txt
23482: 083_pas_soundbox_tones.txt
23483: 084_pas_waves.txt
23484: 085_mxsinus_logo.txt
23485: 085_sinus_plot_waves.txt
23486: 086_pas_graph_arrow_heart.txt
23487: 087_bitmap_loader.txt
23488: 087_pas_bitmap_solution.txt
23489: 087_pas_bitmap_solution2.txt
23490: 087_pas_bitmap_subimage.txt
23491: 087_pas_bitmap_test.txt
23492: 088_pas_soundbox2_mp3.txt
23493: 088_pas_soundbox_mp3.txt
23494: 088_pas_sphere_2.txt
23495: 089_pas_gradient.txt
23496: 089_pas_maxland2.txt
23497: 090_pas_sudoku4.txt
23498: 090_pas_sudoku4_2.txt
23499: 091_pas_cube4.txt
23500: 092_pas_statistics4.txt
23501: 093_variance.txt
23502: 093_variance_debug.txt
23503: 094_pas_daysold.txt
23504: 094_pas_stat_date.txt
23505: 095_pas_ki_simulation.txt
23506: 096_pas_geisen_problem.txt
23507: 096_pas_montyhall_problem.txt
23508: 097_lotto_proofofconcept.txt
23509: 097_pas_lottocombinations_beat_plus.txt
23510: 097_pas_lottocombinations_beat_plus2.txt
23511: 097_pas_lottocombinations_universal.txt
23512: 097_pas_lottosimulation.txt
23513: 098_pas_chartgenerator_plus.txt
23514: 099_pas_3D_show.txt
23515: 200_big_numbers.txt
23516: 200_big_numbers2.txt
23517: 201_streamload_xml.txt
23518: 202_systemcheck.txt
23519: 203_webservice_simple_intftester.txt
23520: 204_webservice_simple.txt
23521: 205_future_value_service.txt
23522: 206_DTD_string_functions.txt
23523: 207_ibz2_async_process.txt
23524: 208_crc32_hash.txt
23525: 209_cryptohash.txt
23526: 210_public_private.txt
23527: 210_public_private_cryptosystem.txt
23528: 211_wipe_pattern.txt
23529: 211_wipe_pattern2.txt
23530: 211_wipe_pattern_solution.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
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

23531: 212_pas_statisticmodule4.TXT
23532: 212_pas_statisticmoduletxt.TXT
23533: 212_statisticmodule4.txt
23534: 213_pas_BBP_Algo.txt
23535: 214_mxdocudemo.txt
23536: 214_mxdocudemo2.txt
23537: 214_mxdocudemo3.txt
23538: 215_hints_test.TXT
23539: 216_warnings_test.TXT
23540: 217_pas_heartbeat.txt
23541: 218_biorythmus_studio.txt
23542: 219_cipherbox.txt
23543: 219_crypt_source_comtest_solution.TXT
23544: 220_cipherbox_form.txt
23545: 220_cipherbox_form2.txt
23546: 221_bcd_explain.txt
23547: 222_memoform.txt
23548: 223_directorybox.txt
23549: 224_dialogs.txt
23550: 225_sprite_animation.txt
23551: 226_ASCII_Grid2.TXT
23552: 227_animation.txt
23553: 227_animation2.txt
23554: 228_android_calendar.txt
23555: 229_android_game.txt
23556: 229_android_game_tester.txt
23557: 230_DataProvider.txt
23558: 230_DataSetProvider.txt
23559: 230_DataSetXMLBackupScholz.txt
23560: 231_DBGrid_access.txt
23561: 231_DBGrid_XMLaccess.txt
23562: 231_DBGrid_XMLaccess2.txt
23563: 231_DBGrid_XMLaccess_locatetester.txt
23564: 231_DBGrid_XML_CDS_local.txt
23565: 232_outline.txt
23566: 232_outline_2.txt
23567: 233_modular_form.txt
23568: 234_debugoutform.txt
23569: 235_fastform.TXT
23570: 236_componentpower.txt
23571: 236_componentpower_back.txt
23572: 237_pas_4forms.txt
23573: 238_lottogen_form.txt
23574: 239_pas_sierpinski.txt
23575: 239_pas_sierpinski2.txt
23576: 240_unitGlobal_tester.txt
23577: 241_db3_sql_tutorial.txt
23578: 241_db3_sql_tutorial2.txt
23579: 241_db3_sql_tutorial2fix.txt
23580: 241_db3_sql_tutorial3.txt
23581: 241_db3_sql_tutorial3connect.txt
23582: 241_db3_sql_tutorial3_fptest.txt
23583: 241_RTL_SET2.txt
23584: 241_RTL_SET2_tester.txt
23585: 242_Component_Control.txt
23586: 243_tutorial_loader.txt
23587: 244_script_loader_loop.txt
23588: 245_formapp2.txt
23589: 245_formapp2_tester.txt
23590: 245_formapp2_testerX.txt
23591: 246_httpapp.txt
23592: 247_datecalendar.txt
23593: 248_ASCII_Grid2_sorted.TXT
23594: 249_picture_grid.TXT
23595: 250_tipsandtricks2.txt
23596: 250_tipsandtricks3.txt
23597: 250_tipsandtricks3api.txt
23598: 250_tipsandtricks3_admin_elevation.txt
23599: 250_tipsandtricks3_tester.txt
23600: 250_tipsandtricks4_tester.txt
23601: 250_tipsandtricks4_tester2.txt
23602: 251_compare_noise_gauss.txt
23603: 251_whitenoise.txt
23604: 251_whitenoise2.txt
23605: 252_hilbert_turtle.txt
23606: 252_pas_hilbert.txt
23607: 253_opearatingsystem3.txt
23608: 254_dynarrays.txt
23609: 255_einstein.txt
23610: 256_findconsts_of_EXE.txt
23611: 256_findfunctions2_of_EXE.txt
23612: 256_findfunctions2_of_EXEaverp.txt
23613: 256_findfunctions2_of_EXEspec.txt
23614: 256_findfunctions3.txt
23615: 256_findfunctions_of_EXE.txt
23616: 257_AES_Cipher.txt
23617: 258_AES_cryptobox.txt
23618: 258_AES_cryptobox2.txt
23619: 258_AES_cryptobox2_passdlg.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
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

```
23620: 259_AES_crypt_directory.txt
23621: 260_sendmessage_2.TXT
23622: 260_sendmessage_beta.TXT
23623: 261_probability.txt
23624: 262_mxoutputdemo4.txt
23625: 263_async_sound.txt
23626: 264_vclutils.txt
23627: 264_VCL_utils2.txt
23628: 265_timer_API.txt
23629: 266_serial_interface.txt
23630: 266_serial_interface2.txt
23631: 266_serial_interface3.txt
23632: 267_ackermann_rec.txt
23633: 267_ackermann_variants.txt
23634: 268_DBGrid_tree.txt
23635: 269_record_grid.TXT
23636: 270_Jedi_FunctionPower.txt
23637: 270_Jedi_FunctionPowertester.txt
23638: 271_closures_study.txt
23639: 271_closures_study_workingset2.txt
23640: 272_pas_function_show.txt
23641: 273_pas_function_show2.txt
23642: 274_library_functions.txt
23643: 275_turtle_language.txt
23644: 275_turtle_language_save.txt
23645: 276_save_algo.txt
23646: 276_save_algo2.txt
23647: 277_functionsfor39.txt
23648: 278_DB_Dialogs.TXT
23649: 279_hexer2.TXT
23650: 279_hexer2macro.TXT
23651: 279_hexer2macroback.TXT
23652: 280_UML_process.txt
23653: 280_UML_process_knabe2.txt
23654: 280_UML_process_knabe3.txt
23655: 280_UML_process_TIM_Botzenhardt.txt
23656: 280_UML_TIM_Seitz.txt
23657: 281_picturepuzzle.txt
23658: 281_picturepuzzle2.txt
23659: 281_picturepuzzle3.txt
23660: 281_picturepuzzle4.txt
23661: 479_inputquery.txt
23662: 480_regex_pathfinder2.txt
23663: 482 processPipe.txt
23664: 483_PathFuncTest_mX.txt
23665: 485_InnoFunc.txt
23666: 487_asyncKeyState.txt
23667: 489_simpleComport.txt
23668: 491_analogmeter.txt
23669: 493_gadgets.txt
23670: 496_InstallX.txt
23671: 498_UnitTesting.txt
23672: 500_diceoflifes.txt
23673: 502_findalldocs.txt
23674: 504_fileclass.txt
23675: 506_colormatrix.txt
23676: 508_simplecomportmorse.txt
23677: 509_509_GEOMap2_SReverse.TXT
23678: 511_LEDLabel.txt
23679: 513_StreamIntegration.txt
23680: 515_ledclock3.txt
23681: 517_animation7.txt
23682: 519_powtils.txt
23683: 521_iputils2.txt
23684: 523_NMEA.txt
23685: 525_GEO84s.txt
23686: 527_GPSDemo.txt
23687: 529_profilertest.txt
23688: 531_profilertest.txt
23689: 533_syncasync_demo.txt
23690: 535_Battleship3.pas
23691: 537_iniplus.TXT
23692: 539_timeturtle123.txt
23693: 541_webserver_arduino_motorturtle.txt
23694: 543_MATH_TurboP.PAS
23695: 545_strips.TXT
23696: 547_regexmaster.TXT
23697: 549_3D_Panorama.txt
23698: 551_ArduinoTester.txt
23699: 553_ArduinoCockBit3.txt
23700: 556_stringlistrandom.TXT
23701: 558_highrestimer.TXT  -559
23702:
23703:
23704: Web Script Examples:
23705: http://www.softwareschule.ch/examples/performer.txt';
23706: http://www.softwareschule.ch/examples/turtle.txt';
23707: http://www.softwareschule.ch/examples/SQLExport.txt';
23708: http://www.softwareschule.ch/examples/Richter.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
484_filefinder3.txt
486_VideoGrabber.txt
488_asyncTerminal.txt
490_webCamproc.txt
492_snowflake2.txt
495_fourierfreq.txt
497_LED.txt
499_mulu42.txt
501_firebird_datasnap_tests.txt
503_led_switch.txt
505_debug.txt
507_derutils.txt
509_GEOMap2.txt
510_510_bonn_gpsdata_mX4.pas
512_LED_moon.txt
514_LED_moon2.txt
516_mapview.txt
518_sensors_meter.txt
520_run_bytecode.txt
522_getgeocode.txt
524_NAV_Utils.txt
526_Compass_meter.txt
528_linescount.txt
530_3DLab.txt
532_mcicommand.txt
534_arduino_cockpit.TXT
536_ressource_grid2.txt
538_shellbatch.txt
540_NeuralNetwork.pas
542_arduino_sound.txt
544_UTIL01.PAS
546_fourier3.pas
548_STExpressions.TXT
550_Expressions.TXT - 550_ADO_OLEDB.txt
552_WaitExec32.txt
554_Watdchdog.txt -   555_CODEsign2.txt
557_4dice2015.txt
560_PSUtils.TXT
```

```
23709: http://www.softwareschule.ch/examples/checker.txt';
23710: http://www.softwareschule.ch/examples/democript.txt';
23711: http://www.softwareschule.ch/examples/ibzresult.txt';
23712: http://www.softwareschule.ch/examples/performindex.txt
23713: http://www.softwareschule.ch/examples/processlist.txt
23714: http://www.softwareschule.ch/examples/game.txt
23715: http://www.softwareschule.ch/examples/GEOGPS.txt
23716: http://www.softwareschule.ch/examples/turtle2.txt
23717: http://www.softwareschule.ch/examples/asyncterminal.txt
23718: http://www.softwareschule.ch/examples/snowflake.txt'
23719: http://www.softwareschule.ch/examples/arduinoled.txt
23720:
23721:
23722: Delphi Basics Run Time Library listing
23723: ********************************************************************************
23724: A
23725: Compiler Directive $A Determines whether data is aligned or packed
23726: Compiler Directive $Align Determines whether data is aligned or packed
23727: Compiler Directive $AppType Determines the application type : GUI or Console
23728: Procedure SysUtils Abort Aborts the current processing with a silent exception
23729: Function System Abs Gives the absolute value of a number (-ve sign is removed)
23730: Directive Abstract Defines a class method only implemented in subclasses
23731: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
23732: Function System Addr Gives the address of a variable, function or procedure
23733: Keyword And Boolean and or bitwise and of two arguments
23734: Type System AnsiChar A character type guaranteed to be 8 bits in size
23735: Function SysUtils AnsiCompareStr Compare two strings for equality
23736: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
23737: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
23738: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
23739: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
23740: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
23741: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
23742: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
23743: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
23744: Function StrUtils AnsiPos Find the position of one string in another
23745: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
23746: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
23747: Function StrUtils AnsiRightStr Extracts characters from the right of a string
23748: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
23749: Type System AnsiString A data type that holds a string of AnsiChars
23750: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
23751: Procedure System Append Open a text file to allow appending of text to the end
23752: Procedure SysUtils AppendStr Concatenate one string onto the end of another
23753: Function Math ArcCos The Arc Cosine of a number, returned in radians
23754: Function Math ArcSin The Arc Sine of a number, returned in radians
23755: Function System ArcTan The Arc Tangent of a number, returned in radians
23756: Keyword Array A data type holding indexable collections of data
23757: Keyword As Used for casting object references
23758: Procedure System Assign Assigns a file handle to a binary or text file
23759: Function System Assigned Returns true if a reference is not nil
23760: Procedure System AssignFile Assigns a file handle to a binary or text file
23761: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
23762:
23763: B
23764: Compiler Directive $B Whether to short cut and and or operations
23765: Compiler Directive $BoolEval Whether to short cut and and or operations
23766: Procedure SysUtils Beep Make a beep sound
23767: Keyword Begin Keyword that starts a statement block
23768: Function System BeginThread Begins a separate thread of code execution
23769: Procedure System BlockRead Reads a block of data records from an untyped binary file
23770: Procedure System BlockWrite Writes a block of data records to an untyped binary file
23771: Type System Boolean Allows just True and False values
23772: Function Classes Bounds Create a TRect value from top left and size values
23773: Procedure System Break Forces a jump out of a single loop
23774: Type System Byte An integer type supporting values 0 to 255
23775:
23776: C
23777: Type System Cardinal The basic unsigned integer type
23778: Keyword Case A mechanism for acting upon different values of an Ordinal
23779: Function StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
23780: Function SysUtils ChangeFileExt Change the extension part of a file name
23781: Type System Char Variable type holding a single character
23782: Procedure System ChDir Change the working drive plus path for a specified drive
23783: Function System Chr Convert an integer into a character
23784: Keyword Class Starts the declaration of a type of object class
23785: Procedure System Close Closes an open file
23786: Procedure System CloseFile Closes an open file
23787: Variable System CmdLine Holds the execution text used to start the current program
23788: Type System Comp A 64 bit signed integer
23789: Function SysUtils CompareStr Compare two strings to see which is greater than the other
23790: Function SysUtils CompareText Compare two strings for equality, ignoring case
23791: Function Math CompareValue Compare numeric values with a tolerance
23792: Function System Concat Concatenates one or more strings into one string
23793: Keyword Const Starts the definition of fixed data values
23794: Keyword Constructor Defines the method used to create an object from a class
23795: Procedure System Continue Forces a jump to the next iteration of a loop
23796: Function ConvUtils Convert Convert one measurement value to another
23797: Function System Copy Create a copy of part of a string or an array
```

```
23798: Function System Cos The Cosine of a number
23799: Function SysUtils CreateDir Create a directory
23800: Type System Currency A floating point type with 4 decimals used for financial values
23801: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
23802: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
23803: Variable SysUtils CurrencyString The currency string used in currency display functions
23804: Function SysUtils CurrToStr Convert a currency value to a string
23805: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
23806:
23807: D
23808: Compiler Directive $D Determines whether application debug information is built
23809: Compiler Directive $DebugInfo Determines whether application debug information is built
23810: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
23811: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
23812: Function SysUtils Date Gives the current date
23813: Variable SysUtils DateSeparator The character used to separate display date fields
23814: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
23815: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
23816: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
23817: Function SysUtils DateToStr Converts a TDateTime date value to a string
23818: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
23819: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
23820: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
23821: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
23822: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
23823: Function DateUtils DaysInAMonth Gives the number of days in a month
23824: Function DateUtils DaysInAYear Gives the number of days in a year
23825: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
23826: Procedure System Dec Decrement an ordinal variable
23827: Variable SysUtils DecimalSeparator The character used to display the decimal point
23828: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
23829: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
23830: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
23831: Directive Default Defines default processing for a property
23832: Function Math DegToRad Convert a degrees value to radians
23833: Procedure System Delete Delete a section of characters from a string
23834: Function SysUtils DeleteFile Delete a file specified by its file name
23835: Keyword Destructor Defines the method used to destroy an object
23836: Function SysUtils DirectoryExists Returns true if the given directory exists
23837: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
23838: Function SysUtils DiskSize Gives the size in bytes of a specified drive
23839: Procedure System Dispose Dispose of storage used by a pointer type variable
23840: Keyword Div Performs integer division, discarding the remainder
23841: Keyword Do Defines the start of some controlled action
23842: Type System Double A floating point type supporting about 15 digits of precision
23843: Keyword DownTo Prefixes an decremental for loop target value
23844: Function StrUtils DupeString Creates a string containing copies of a substring
23845: Directive Dynamic Allows a class method to be overriden in derived classes
23846:
23847: E
23848: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
23849: Compiler Directive $EndIf Terminates conditional code compilation
23850: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
23851: Keyword Else Starts false section of if, case and try statements
23852: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
23853: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
23854: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
23855: Keyword End Keyword that terminates statement blocks
23856: Function DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
23857: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
23858: Procedure System EndThread Terminates a thread with an exit code
23859: Function System Eof Returns true if a file opened with Reset is at the end
23860: Function System Eoln Returns true if the current text file is pointing at a line end
23861: Procedure System Erase Erase a file
23862: Variable System ErrorAddr Sets the error address when an application terminates
23863: Keyword Except Starts the error trapping clause of a Try statement
23864: Procedure System Exclude Exclude a value in a set variable
23865: Procedure System Exit Exit abruptly from a function or procedure
23866: Variable System ExitCode Sets the return code when an application terminates
23867: Function System Exp Gives the exponent of a number
23868: Directive System Export Makes a function or procedure in a DLL externally available
23869: Type System Extended The floating point type with the highest capacity and precision
23870: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
23871: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
23872: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
23873: Function SysUtils ExtractFileName Extracts the name part of a full file name
23874: Function SysUtils ExtractFilePath Extracts the path part of a full file name
23875:
23876: F
23877: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
23878: Keyword File Defines a typed or untyped file
23879: Function SysUtils FileAge Get the last modified date/time of a file without opening it
23880: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
23881: Function SysUtils FileExists Returns true if the given file exists
23882: Function SysUtils FileGetAttr Gets the attributes of a file
23883: Variable System FileMode Defines how Reset opens a binary file
23884: Function System FilePos Gives the file position in a binary or text file
23885: Function SysUtils FileSearch Search for a file in one or more directories
23886: Function SysUtils FileSetAttr Sets the attributes of a file
```

```
23887: Function SysUtils FileSetDate Set the last modified date and time of a file
23888: Function System FileSize Gives the size in records of an open file
23889: Procedure System FillChar Fills out a section of storage with a fill character or byte value
23890: Keyword Finally Starts the unconditional code section of a Try statement
23891: Function SysUtils FindClose Closes a successful FindFirst file search
23892: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
23893: Function SysUtils FindFirst Finds all files matching a file mask and attributes
23894: Function SysUtils FindNext Find the next file after a successful FindFirst
23895: Function SysUtils FloatToStr Convert a floating point value to a string
23896: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
23897: Procedure System Flush Flushes buffered text file data to the file
23898: Keyword For Starts a loop that executes a finite number of times
23899: Function SysUtils ForceDirectories Create a new path of directories
23900: Function SysUtils Format Rich formatting of numbers and text into a string
23901: Function SysUtils FormatCurr Rich formatting of a currency value into a string
23902: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
23903: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
23904: Function System Frac The fractional part of a floating point number
23905: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
23906: Procedure System FreeMem Free memory storage used by a variable
23907: Keyword System Function Defines a subroutine that returns a value
23908:
23909: G
23910: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
23911: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
23912: Function System GetLastError Gives the error code of the last failing Windows API call
23913: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
23914: Function System GetMem Get a specified number of storage bytes
23915: Keyword Goto Forces a jump to a label, regardless of nesting
23916:
23917: H
23918: Compiler Directive $H Treat string types as AnsiString or ShortString
23919: Compiler Directive $Hints Determines whether Delphi shows compilation hints
23920: Procedure System Halt Terminates the program with an optional dialog
23921: Function System Hi Returns the hi-order byte of a (2 byte) Integer
23922: Function System High Returns the highest value of a type or variable
23923:
23924: I
23925: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
23926: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
23927: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
23928: Compiler Directive $IfOpt Tests for the state of a Compiler directive
23929: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
23930: Compiler Directive $IOChecks When on, an IO operation error throws an exception
23931: Keyword If Starts a conditional expression to determine what to do next
23932: Keyword Implementation Starts the implementation (code) section of a Unit
23933: Keyword In Used to test if a value is a member of a set
23934: Procedure System Inc Increment an ordinal variable
23935: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
23936: Procedure System Include Include a value in a set variable
23937: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
23938: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
23939: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
23940: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
23941: Function DateUtils IncYear Increments a TDateTime variable by a number of years
23942: Directive Index Principally defines indexed class data properties
23943: Constant Math Infinity Floating point value of infinite size
23944: Keyword Inherited Used to call the parent class constructor or destructor method
23945: Variable System Input Defines the standard input text file
23946: Function Dialogs InputBox Display a dialog that asks for user text input, with default
23947: Function Dialogs InputQuery Display a dialog that asks for user text input
23948: Procedure System Insert Insert a string into another string
23949: Function System Int The integer part of a floating point number as a float
23950: Type System Int64 A 64 bit sized integer - the largest in Delphi
23951: Type System Integer The basic Integer type
23952: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
23953: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
23954: Function SysUtils IntToStr Convert an integer into a string
23955: Function System IOResult Holds the return code of the last I/O operation
23956: Keyword Is Tests whether an object is a certain class or ascendant
23957: Function Math IsInfinite Checks whether a floating point number is infinite
23958: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
23959: Function System IsMultiThread Returns true if the code is running multiple threads
23960: Function Math IsNaN Checks to see if a floating point number holds a real number
23961:
23962: L
23963: Compiler Directive $L Determines what application debug information is built
23964: Compiler Directive $LocalSymbols Determines what application debug information is built
23965: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
23966: Function SysUtils LastDelimiter Find the last position of selected characters in a string
23967: Function System Length Return the number of elements in an array or string
23968: Function System Ln Gives the natural logarithm of a number
23969: Function System Lo Returns the low-order byte of a (2 byte) Integer
23970: Function Math Log10 Gives the log to base 10 of a number
23971: Variable SysUtils LongDateFormat Long version of the date to string format
23972: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
23973: Type System LongInt An Integer whose size is guaranteed to be 32 bits
23974: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
23975: Variable SysUtils LongTimeFormat Long version of the time to string format
```

```
23976: Type System LongWord A 32 bit unsigned integer
23977: Function System Low Returns the lowest value of a type or variable
23978: Function SysUtils LowerCase Change upper case characters in a string to lower case
23979:
23980: M
23981: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
23982: Function Math Max Gives the maximum of two integer values
23983: Constant System MaxInt The maximum value an Integer can have
23984: Constant System MaxLongInt The maximum value an LongInt can have
23985: Function Math Mean Gives the average for a set of numbers
23986: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
23987: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
23988: Function Math Min Gives the minimum of two integer values
23989: Constant SysUtils MinsPerDay Gives the number of minutes in a day
23990: Procedure System MkDir Make a directory
23991: Keyword Mod Performs integer division, returning the remainder
23992: Constant SysUtils MonthDays Gives the number of days in a month
23993: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
23994: Procedure System Move Copy bytes of data from a source to a destination
23995:
23996: N
23997: Constant Math NaN Not a real number
23998: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
23999: Procedure System New Create a new pointer type variable
24000: Constant System Nil A pointer value that is defined as undetermined
24001: Keyword Not Boolean Not or bitwise not of one arguments
24002: Function SysUtils Now Gives the current date and time
24003: Variable Variants Null A variable that has no value
24004:
24005: O
24006: Compiler Directive $O Determines whether Delphi optimises code when compiling
24007: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
24008: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
24009: Keyword System Object Allows a subroutine data type to refer to an object method
24010: Function System Odd Tests whether an integer has an odd value
24011: Keyword Of Linking keyword used in many places
24012: Keyword On Defines exception handling in a Try Except clause
24013: Keyword Or Boolean or or bitwise or of two arguments
24014: Function System Ord Provides the Ordinal value of an integer, character or enum
24015: Directive Out Identifies a routine parameter for output only
24016: Variable System Output Defines the standard output text file
24017: Directive Overload Allows 2 or more routines to have the same name
24018: Directive Override Defines a method that replaces a virtual parent class method
24019:
24020: P
24021: Keyword Packed Compacts complex data types into minimal storage
24022: Type System PAnsiChar A pointer to an AnsiChar value
24023: Type System PAnsiString Pointer to an AnsiString value
24024: Function System ParamCount Gives the number of parameters passed to the current program
24025: Function System ParamStr Returns one of the parameters used to run the current program
24026: Type System PChar A pointer to an Char value
24027: Type System PCurrency Pointer to a Currency value
24028: Type System PDateTime Pointer to a TDateTime value
24029: Type System PExtended Pointer to a Extended floating point value
24030: Function System Pi The mathematical constant
24031: Type System PInt64 Pointer to an Int64 value
24032: Function Classes Point Generates a TPoint value from X and Y values
24033: Type System Pointer Defines a general use Pointer to any memory based data
24034: Function Classes PointsEqual Compares two TPoint values for equality
24035: Function System Pos Find the position of one string in another
24036: Function System Pred Decrement an ordinal variable
24037: Function Printers Printer Returns a reference to the global Printer object
24038: Directive Private Starts the section of private data and methods in a class
24039: Keyword System Procedure Defines a subroutine that does not return a value
24040: Procedure FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
24041: Keyword System Program Defines the start of an application
24042: Function Dialogs PromptForFileName Shows a dialog allowing the user to select a file
24043: Keyword System Property Defines controlled access to class fields
24044: Directive Protected Starts a section of class private data accesible to sub-classes
24045: Type System PShortString A pointer to an ShortString value
24046: Type System PString Pointer to a String value
24047: Function Types PtInRect Tests to see if a point lies within a rectangle
24048: Directive Public Starts an externally accessible section of a class
24049: Directive Published Starts a published externally accessible section of a class
24050: Type System PVariant Pointer to a Variant value
24051: Type System PWideChar Pointer to a WideChar
24052: Type System PWideString Pointer to a WideString value
24053:
24054: Q
24055: Compiler Directive $Q Determines whether Delphi checks integer and enum bounds
24056:
24057: R
24058: Compiler Directive $R Determines whether Delphi checks array bounds
24059: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
24060: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
24061: Compiler Directive $Resource Defines a resource file to be included in the application linking
24062: Function Math RadToDeg Converts a radian value to degrees
24063: Keyword Raise Raise an exception
24064: Function System Random Generate a random floating point or integer number
```

```
24065: Procedure System Randomize Reposition the Random number generator next value
24066: Function Math RandomRange Generate a random integer number within a supplied range
24067: Variable System RandSeed Reposition the Random number generator next value
24068: Procedure System Read Read data from a binary or text file
24069: Procedure System ReadLn Read a complete line of data from a text file
24070: Type System Real A floating point type supporting about 15 digits of precision
24071: Type System Real48 The floating point type with the highest capacity and precision
24072: Procedure System ReallocMem Reallocate an existing block of storage
24073: Function DateUtils RecodeDate Change only the date part of a TDateTime variable
24074: Function DateUtils RecodeTime Change only the time part of a TDateTime variable
24075: Keyword Record A structured data type - holding fields of data
24076: Function Classes Rect Create a TRect value from 2 points or 4 coordinates
24077: Function SysUtils RemoveDir Remove a directory
24078: Procedure System Rename Rename a file
24079: Function SysUtils RenameFile Rename a file or directory
24080: Keyword Repeat Repeat statements until a ternmination condition is met
24081: Procedure SysUtils ReplaceDate Change only the date part of a TDateTime variable
24082: Procedure SysUtils ReplaceTime Change only the time part of a TDateTime variable
24083: Procedure System Reset Open a text file for reading, or binary file for read/write
24084: Variable System Result A variable used to hold the return value from a function
24085: Procedure System ReWrite Open a text or binary file for write access
24086: Procedure System RmDir Remove a directory
24087: Function System Round Rounds a floating point number to an integer
24088: Procedure System RunError Terminates the program with an error dialog
24089:
24090: S
24091: Constant SysUtils SecsPerDay Gives the number of seconds in a day
24092: Procedure System Seek Move the pointer in a binary file to a new record position
24093: Function System SeekEof Skip to the end of the current line or file
24094: Function System SeekEoln Skip to the end of the current line or file
24095: Function FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
24096: Variable System Self Hidden parameter to a method - refers to the containing object
24097: Keyword Set Defines a set of up to 255 distinct values
24098: Function SysUtils SetCurrentDir Change the current directory
24099: Procedure System SetLength Changes the size of a string, or the size(s) of an array
24100: Procedure System SetString Copies characters from a buffer into a string
24101: Keyword Shl Shift an integer value left by a number of bits
24102: Variable SysUtils ShortDateFormat Compact version of the date to string format
24103: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
24104: Type System ShortInt An integer type supporting values -128 to 127
24105: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
24106: Type System ShortString Defines a string of up to 255 characters
24107: Variable SysUtils ShortTimeFormat Short version of the time to string format
24108: Procedure Dialogs ShowMessage Display a string in a simple dialog with an OK button
24109: Procedure Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
24110: Procedure Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
24111: Keyword Shr Shift an integer value right by a number of bits
24112: Function System Sin The Sine of a number
24113: Type System Single The smallest capacity and precision floating point type
24114: Function System SizeOf Gives the storage byte size of a type or variable
24115: Function System Slice Creates a slice of an array as an Open Array parameter
24116: Type System SmallInt An Integer type supporting values from -32768 to 32767
24117: Function System Sqr Gives the square of a number
24118: Function System Sqrt Gives the square root of a number
24119: Procedure System Str Converts an integer or floating point number to a string
24120: Type System String A data type that holds a string of characters
24121: Function System StringOfChar Creates a string with one character repeated many times
24122: Function SysUtils StringReplace Replace one or more substrings found within a string
24123: Function System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
24124: Function SysUtils StrScan Searches for a specific character in a constant string
24125: Function SysUtils StrToCurr Convert a number string into a currency value
24126: Function SysUtils StrToDate Converts a date string into a TDateTime value
24127: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
24128: Function SysUtils StrToFloat Convert a number string into a floating point value
24129: Function SysUtils StrToInt Convert an integer string into an Integer value
24130: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
24131: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
24132: Function SysUtils StrToIntDef Convert a string into an Integer value with default
24133: Function SysUtils StrToTime Converts a time string into a TDateTime value
24134: Function StrUtils StuffString Replaces a part of one string with another
24135: Function System Succ Increment an ordinal variable
24136: Function Math Sum Return the sum of an array of floating point values
24137:
24138: T
24139: Function Math Tan The Tangent of a number
24140: Type Classes TBits An object that can hold an infinite number of Boolean values
24141: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
24142: Type ConvUtils TConvType Defines a measurement type as used by Convert
24143: Type System TDateTime Data type holding a date and time value
24144: Type System Text Defines a file as a text file
24145: Type System TextFile Declares a file type for storing lines of text
24146: Type SysUtils TFloatFormat Formats for use in floating point number display functions
24147: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
24148: Keyword Then Part of an if statement - starts the true clause
24149: Variable SysUtils ThousandSeparator The character used to display the thousands separator
24150: Keyword ThreadVar Defines variables that are given separate instances per thread
24151: Function SysUtils Time Gives the current time
24152: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
24153: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
```

```
24154: Variable SysUtils TimeSeparator The character used to separate display time fields
24155: Function SysUtils TimeToStr Converts a TDateTime time value to a string
24156: Type Classes TList General purpose container of a list of objects
24157: Keyword To Prefixes an incremental for loop target value
24158: Type System TObject The base class type that is ancestor to all other classes
24159: Function DateUtils Tomorrow Gives the date tomorrow
24160: Type Dialogs TOpenDialog Displays a file selection dialog
24161: Type Types TPoint Holds X and Y integer values
24162: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
24163: Type Types TRect Holds rectangle coordinate values
24164: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
24165: Function SysUtils Trim Removes leading and trailing blanks from a string
24166: Function SysUtils TrimLeft Removes leading blanks from a string
24167: Function SysUtils TrimRight Removes trailing blanks from a string
24168: Function System Trunc The integer part of a floating point number
24169: Procedure System Truncate Truncates a file size - removes all data after the current position
24170: Keyword Try Starts code that has error trapping
24171: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
24172: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
24173: Type Classes TStringList Holds a variable length list of strings
24174: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
24175: Type System TThreadFunc Defines the function to be called by BeginThread
24176: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
24177: Keyword Type Defines a new category of variable or process
24178:
24179: U
24180: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
24181: Keyword Unit Defines the start of a unit file - a Delphi module
24182: Keyword Until Ends a Repeat control loop
24183: Function System UpCase Convert a Char value to upper case
24184: Function SysUtils UpperCase Change lower case characters in a string to upper case
24185: Keyword Uses Declares a list of Units to be imported
24186:
24187: V
24188: Procedure System Val Converts number strings to integer and floating point values
24189: Keyword Var Starts the definition of a section of data variables
24190: Type System Variant A variable type that can hold changing data types
24191: Function Variants VarType Gives the current type of a Variant variable
24192: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
24193: Directive Virtual Allows a class method to be overriden in derived classes
24194:
24195: W
24196: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
24197: Keyword While Repeat statements whilst a continuation condition is met
24198: Type System WideChar Variable type holding a single International character
24199: Function System WideCharToString Copies a null terminated WideChar string to a normal string
24200: Type System WideString A data type that holds a string of WideChars
24201: Keyword With A means of simplifying references to structured variables
24202: Type System Word An integer type supporting values 0 to 65535
24203: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
24204: Procedure System Write Write data to a binary or text file
24205: Procedure System WriteLn Write a complete line of data to a text file
24206:
24207: X
24208: Compiler Directive $X Controls some Pascal extension handling
24209: Keyword Xor Boolean Xor or bitwise Xor of two arguments
24210:
24211: Y
24212: Compiler Directive $Y Determines whether application symbol information is built
24213: Function DateUtils Yesterday Gives the date yesterday
24214:
24215: Z
24216: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
24217:
24218: procedure SIRegister_uPSUtils(CL: TPSPascalCompiler);
24219: begin
24220:  PSMainProcName','String').SetString( '!MAIN');
24221:  PSMainProcNameOrg','String').SetString( 'Main Proc');
24222:  'PSLowBuildSupport','LongInt').SetInt( 12);
24223:  CL.AddConstantN('PSCurrentBuildNo','LongInt').SetInt( 23);
24224:  'PSCurrentversion','String').SetString( '1.31');
24225:  'PSValidHeader','LongInt').SetInt( 1397769801);
24226:  'PSAddrStackStart','LongInt').SetInt( 1610612736);
24227:  'PSAddrNegativeStackStart','LongInt').SetInt( 1073741824);
24228:  //'TPSBaseType','').SetString( Byte);
24229:  'btReturnAddress','LongInt').SetInt( 0);
24230:  'btU8','LongInt').SetInt( 1);
24231:  'btS8','LongInt').SetInt( 2);
24232:  'btU16','LongInt').SetInt( 3);
24233:  'btS16','LongInt').SetInt( 4);
24234:  'btU32','LongInt').SetInt( 5);
24235:  'btS32','LongInt').SetInt( 6);
24236:  'btSingle','LongInt').SetInt( 7);
24237:  'btDouble','LongInt').SetInt( 8);
24238:  'btExtended','LongInt').SetInt( 9);
24239:  'btString','LongInt').SetInt( 10);
24240:  'btRecord','LongInt').SetInt( 11);
24241:  'btArray','LongInt').SetInt( 12);
24242:  'btPointer','LongInt').SetInt( 13);
```

```
24243:  'btPChar','LongInt').SetInt( 14);
24244:  'btResourcePointer','LongInt').SetInt( 15);
24245:  'btVariant','LongInt').SetInt( 16);
24246:  'btS64','LongInt').SetInt( 17);
24247:  'btU64','LongInt').SetInt( 30);
24248:  'btChar','LongInt').SetInt( 18);
24249:  'btWideString','LongInt').SetInt( 19);
24250:  'btWideChar','LongInt').SetInt( 20);
24251:  'btProcPtr','LongInt').SetInt( 21);
24252:  'btStaticArray','LongInt').SetInt( 22);
24253:  'btSet','LongInt').SetInt( 23);
24254:  'btCurrency','LongInt').SetInt( 24);
24255:  'btClass','LongInt').SetInt( 25);
24256:  'btInterface','LongInt').SetInt( 26);
24257:  'btNotificationVariant','LongInt').SetInt( 27);
24258:  'btUnicodeString','LongInt').SetInt( 28);
24259:  'btType','LongInt').SetInt( 130);
24260:  'btEnum','LongInt').SetInt( 129);
24261:  'btExtClass','LongInt').SetInt( 131);
24262: // CL.AddDelphiFunction('Function MakeHash( const s : TbtString) : Longint');
24263:  'CM_A','LongInt').SetInt( 0);
24264:  'CM_CA','LongInt').SetInt( 1);
24265:  'CM_P','LongInt').SetInt( 2);
24266:  'CM_PV','LongInt').SetInt( 3);
24267:  'CM_PO','LongInt').SetInt( 4);
24268:  'Cm_C','LongInt').SetInt( 5);
24269:  'Cm_G','LongInt').SetInt( 6);
24270:  'Cm_CG','LongInt').SetInt( 7);
24271:  'Cm_CNG','LongInt').SetInt( 8);
24272:  'Cm_R','LongInt').SetInt( 9);
24273:  'Cm_ST','LongInt').SetInt( 10);
24274:  'Cm_Pt','LongInt').SetInt( 11);
24275:  'CM_CO','LongInt').SetInt( 12);
24276:  'Cm_cv','LongInt').SetInt( 13);
24277:  'cm_sp','LongInt').SetInt( 14);
24278:  'cm_bn','LongInt').SetInt( 15);
24279:  'cm_vm','LongInt').SetInt( 16);
24280:  'cm_sf','LongInt').SetInt( 17);
24281:  'cm_fg','LongInt').SetInt( 18);
24282:  'cm_puexh','LongInt').SetInt( 19);
24283:  'cm_poexh','LongInt').SetInt( 20);
24284:  'cm_in','LongInt').SetInt( 21);
24285:  'cm_spc','LongInt').SetInt( 22);
24286:  'cm_inc','LongInt').SetInt( 23);
24287:  'cm_dec','LongInt').SetInt( 24);
24288:  'cm_nop','LongInt').SetInt( 255);
24289:  'Cm_PG','LongInt').SetInt( 25);
24290:  'Cm_P2G','LongInt').SetInt( 26);
24291:   CL.AddTypeS('TbtU8', 'Byte');
24292:   CL.AddTypeS('TbtS8', 'ShortInt');
24293:   CL.AddTypeS('TbtU16', 'Word');
24294:   CL.AddTypeS('TbtS16', 'SmallInt');
24295:   CL.AddTypeS('TbtU32', 'Cardinal');
24296:   CL.AddTypeS('TbtS32', 'Longint');
24297:   CL.AddTypeS('TbtSingle', 'Single');
24298:   CL.AddTypeS('TbtDouble', 'double');
24299:   CL.AddTypeS('TbtExtended', 'Extended');
24300:   CL.AddTypeS('tbtCurrency', 'Currency');
24301:   CL.AddTypeS('tbts64', 'int64');
24302:   CL.AddTypeS('Tbtu64', 'uint64');
24303:   CL.AddTypeS('TbtString', 'string');
24304:    CL.AddDelphiFunction('Function MakeHash( const s : TbtString) : Longint');
24305: //   TbtString = {$IFDEF DELPHI2009UP}AnsiString{$ELSE}String{$ENDIF};
24306:    //'PointerSize','LongInt').SetInt( IPointer ( 8 4 ));
24307: end;
24308:
24309: ------------------------------------------------------------------------------
24310: mapX:
24311:  if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
24312:     writeln('cologne map found');
24313:     GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
24314:     writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
24315:     OpenMapX('church trier');
24316:     GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
24317:     writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));}
24318:     >>>     //latitude: '50.94133705'  longitude: '6.95812076100766'
24319:     // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
24320:     CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
24321:     Function SendInput( cInputs : UINT; var pInputs : TInput; cbSize : Integer) : UINT');
24322:     Function GetLastInputInfo( var plii : TLastInputInfo) : BOOL');
24323:
24324:
24325: maXbox Ref:
24326: Signature:
24327: SHA1:  maXbox3.exe 8B4D7070BA40BDE17EEDBE78121BB8B474A1D6CF
24328: CRC32: maXbox3.exe 1B550873
24329:
24330: Ref:
24331:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
```

```
24332:     2. shdig: TSHA1Digest;
24333:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
24334:                for i:= 0 to 19 do write(BytetoHex(shdig[i]));
24335:
24336:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
24337:
24338:
24339: https://www.virustotal.
       com/en/file/1623dc4426e09edad749b2a201ba81b3432f57a8a369edd36980c3cd203915d8/analysis/1420818155/
24340:
24341: VirusTotal metadata
24342:
24343: First submission 2015-01-09 15:34:46 UTC ( 10 minutes ago )
24344: Last submission 2015-01-09 15:42:35 UTC ( 2 minutes ago )
24345: File names  Sunprise mX4
24346: maXbox3.exe
24347: maxbox3_9.exe
24348:
24349: SHA256:  1623dc4426e09edad749b2a201ba81b3432f57a8a369edd36980c3cd203915d8
24350: File name:  maXbox3.exe
24351: Detection ratio:  0 / 56
24352: Analysis date:  2015-01-09 15:42:35 UTC ( 2 minutes ago )
24353:
24354: MD5 835f80160705882ada76b74a35b4e42c
24355: SHA1 8b4d7070ba40bde17eedbe78121bb8b474a1d6cf
24356: SHA256 1623dc4426e09edad749b2a201ba81b3432f57a8a369edd36980c3cd203915d8
24357: ssdeep 393216:sDZ2NEY/pE49AOJjCA2ATDFoqvthu0kpC:5+ipE5/QDFb/n
24358:
24359: authentihash 17a6ba597dc043b6f7a903cdc7433a77a59d8a43ae923927658fdaee671927fe
24360: imphash 995a4c7c553245e979876cb44c73c127
24361: File size 22.8 MB ( 23860224 bytes )
24362: File type Win32 EXE
24363: Magic literal
24364: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
24365:
24366: TrID  Windows ActiveX control (36.4%)
24367: Inno Setup installer (34.3%)
24368: InstallShield setup (13.4%)
24369: Win32 EXE PECompact compressed (generic) (13.0%)
24370: Win32 Executable (generic) (1.4%)
24371:
24372:  ExifTool file metadata
24373:  ------------------------
24374:
24375: SpecialBuild mX4
24376: CodeSize 18763776
24377: SubsystemVersion 4.0
24378: Comments reduce to the max
24379: LinkerVersion 2.25
24380: ImageVersion 0.0
24381: FileSubtype 0
24382: FileVersionNumber 3.9.9.160
24383: LanguageCode German (Swiss)
24384: FileFlagsMask 0x003f
24385: FileDescription maXbox Delphi VM
24386: CharacterSet Windows, Latin1
24387: InitializedDataSize 5095424
24388: FileOS Win32
24389: TimeStamp 2015:01:09 13:07:36+01:00
24390: MIMEType application/octet-stream
24391: LegalCopyright Free Pascal Script
24392: FileVersion 3.9.9.160
24393: SpeziellesBuild mX4 Compiler Engine
24394: FileType Win32 EXE
24395: PEType PE32
24396: InternalName Sunprise mX4
24397: FileAccessDate 2015:01:09 16:42:41+01:00
24398: ProductVersion 3.9 Solar mX4
24399: UninitializedDataSize 0
24400: OSVersion 4.0
24401: FileCreateDate 2015:01:09 16:42:41+01:00
24402: OriginalFilename maxbox3_9.exe
24403: Subsystem Windows GUI
24404: MachineType Intel 386 or later, and compatibles
24405: CompanyName kleiner kommunikation
24406: LegalTrademarks maXbox
24407: ProductName maXbox
24408: ProductVersionNumber 3.9.9.160
24409: EntryPoint 0x11e6c7c
24410: ObjectFileType Executable application
24411:
24412:  ExifTool file metadata
24413:  ------------------------
24414: Developer metadata
24415:
24416: Publisher kleiner kommunikation
24417: Product maXbox
24418: Original name maxbox3_9.exe
24419: Internal name Sunprise mX4
```

```
24420: File version 3.9.9.160
24421: Description maXbox Delphi VM
24422: Comments reduce to the max
24423:
24424: PE header basic information
24425:
24426: Target machine Intel 386 or later processors and compatible processors
24427: Compilation timestamp 2015-01-09 12:07:36
24428: Link date 1:07 PM 1/9/2015
24429: Entry Point 0x011E6C7C
24430: Number of sections 10
24431:
24432:
24433: 10 PE sections
24434:
24435: Name Virtual address Virtual size Raw size Entropy MD5
24436:  .text 4096 18715040 18715136 6.60 63b2998c9f3202b17b30b6e46fa00d84
24437: .itext 18722816 48576 48640 6.59 b405dd99d455a1c2a2591e4517d6a897
24438: .data 18771968 265216 265216 5.59 da2b611e8228c925790c856b07b822e5
24439: .bss 19038208 381468 0 0.00 d41d8cd98f00b204e9800998ecf8427e
24440: .idata 19423232 55120 55296 5.67 784b6d3e3180e8e18eb4db7ea79d665e
24441: .edata 19480576 77 512 0.89 1206a6cf24df66a1c8145cb250548a36
24442: .tls 19484672 208 0 0.00 d41d8cd98f00b204e9800998ecf8427e
24443: .rdata 19488768 24 512 0.28 3a08f58c67a6846a10d14ee2b631730e
24444: .reloc 19492864 1191904 1191936 6.76 b8dde920ec8f28b35da2a17ee12c2ab6
24445: .rsrc 20684800 3581952 3581952 5.28 54a4866f7b2ba126c408f9d6e412119b
24446:
24447: PE imports
24448: PE exports CreateIncome
24449:
24450:  Number of PE resources by type
24451:
24452: RT_BITMAP 810
24453: RT_STRING 305
24454: RT_RCDATA 161
24455: RT_ICON 55
24456: RT_GROUP_ICON 44
24457: RT_CURSOR 31
24458: RT_GROUP_CURSOR 26
24459: WAVE 9
24460: RT_DIALOG 2
24461: RT_HTML 2
24462: RT_MESSAGETABLE 1
24463: RT_MANIFEST 1
24464: RT_VERSION 1
24465:
24466:  PE imports
24467:
24468: [+] AVICAP32.DLL  [+] AVICAP32.dll [+] GLU32.dll
24469: [+] IMAGEHLP.DLL  [+] MSVCRT.DLL   [+] MSVFW32.DLL
24470: [+] OpenGL32.dll  [+] SHFolder.dll [+] URLMON.DLL
24471: [+] advapi32.dll  [+] comctl32.dll [+] comdlg32.dll
24472: [+] gdi32.dll     [+] imagehlp.dll [+] imm32.dll
24473: [+] iphlpapi.dll  [+] kernel32.dll [+] mpr.dll
24474: [+] msacm32.dll   [+] ole32.dll    [+] oleacc.dll
24475: [+] oleaut32.dll  [+] oledlg.dll   [+] opengl32.dll
24476: [+] shell32.dll   [+] shlwapi.dll  [+] user32.dll
24477: [+] usp10.dll     [+] version.dll  [+] winhttp.dll
24478: [+] wininet.dll   [+] winmm.dll    [+] winspool.drv
24479: [+] ws2_32.dll    [+] wsock32.dll  [+] msimg32.dll
24480:
24481: Ref:
24482: https://www.virustotal.
       com/en/file/1623dc4426e09edad749b2a201ba81b3432f57a8a369edd36980c3cd203915d8/analysis/1420818155/
24483:
24484: ****************************************************************
24485: Release Notes maXbox 3.9.9.160 January 2015 CODEsign
24486: ****************************************************************
24487: Add 9 Units, 2 Slides 1 Tutor, CLXUp, ExampleEdition, UnitConverter
24488:
24489: 1053 unit uPSI_BigIni                          //Hinzen
24490: 1054 unit uPSI_ShellCtrls;                     //VCL
24491: 1055 unit uPSI_fMath;                          //FMath
24492: 1056 unit uPSI_fComp;                          //FMath
24493: 1057 unit uPSI_HighResTimer;                   //Lauer
24494: 1058 unit uconvMain; (Unit Converter)         //PS
24495: 1059 unit uPSI_uconvMain;                      //PS
24496: 1060 unit uPSI_ParserUtils;                    //PS
24497: 1061 unit uPSI_uPSUtils;                       //PS
24498:
24499:
24500: ****************************************************************
24501: Release Notes maXbox 3.9.9.120 December 2014 CODEsign
24502: ****************************************************************
24503: Add 10 Units, 1Slides, NeuralNetwork, Pan3D View, GDIBackend
24504: 1043 unit uPSI_NeuralNetwork;
24505: 1044 unit uPSI_StExpr;
24506: 1045 unit uPSI_GR32_Geometry;
24507: 1046 unit uPSI_GR32_Containers;
```

```
24508: 1047 unit uPSI_GR32_Backends_VCL,
24509: 1048 unit uPSI_StSaturn; //Venus+Pluto+Mars+Merc+JupSat+++
24510: 1049 unit uPSI_JclParseUses;
24511: 1050 unit uPSI_JvFinalize;
24512: 1051 unit uPSI_panUnit1;
24513: 1052 unit uPSI_DD83u1;  //Arduino Tester
24514:
24515:
24516:     Published Doc maXbox Tutors Starter Introduction 2014
24517:     Tutorial 00 Blix the Programmer
24518:     Tutorial 01 Procedural-Coding
24519:     Tutorial 02 OO-Coding
24520:     Tutorial 03 Modular Coding
24521:     Tutorial 04 UML Coding
24522:     Tutorial 05 Internet Coding
24523:     Tutorial 06 Network Coding
24524:     Tutorial 07 Game Coding
24525:     Tutorial 08 Operating System Coding
24526:     Tutorial 09 Database Coding
24527:     Tutorial 10 Statistic Coding
24528:     Tutorial 11 Forms Coding
24529:     Tutorial 12 SQL Coding
24530:     Tutorial 13 Crypto Coding
24531:     Tutorial 14 Parallel Coding
24532:     Tutorial 15 Serial Coding
24533:     Tutorial 16 Event Driven Coding
24534:     Tutorial 17 Web Server Coding
24535:     Tutorial 18 Arduino Coding and Web of Things
24536:     Tutorial 18_3 Arduino RGB LED Breadboard and Source LED Zip
24537:     Tutorial 19 WinCOM /Arduino Coding and Source LED COM
24538:     Tutorial 20_1 Regular Expressions V2
24539:     Tutorial 21 Android av. End of 2014 and Basta LED Things and Code ADK SeekBar
24540:     Tutorial 22 Services Coding
24541:     Tutorial 23 Real Time Code
24542:     Tutorial 24 Clean Code
24543:     Tutorial 25 Configuration
24544:     Tutorial 26 TCP Sockets
24545:     Tutorial 27 XML Coding
24546:     Tutorial 28 DLL Coding
24547:     Tutorial 29 UML Modeling
24548:     Tutorial 30 WOT Web of Things and Basta 2014 Arduino and maXbox
24549:     Tutorial 31 Closures
24550:     Tutorial 32 SQL Server Firebird
24551:     Tutorial 34 GPS Codes
24552:     Tutorial 35 Web Box
24553:     Tutorial 39 Maps Coding
24554:     Tutorial 39 Maps2 Coding
24555:
24556:
24557:     Published Doc maXbox Example Edition 2015
24558:      \example_edition\01_Algorithm');
24559:      \example_edition\02_Graphics');
24560:      \example_edition\03_Games');
24561:      \example_edition\04_Multimedia');
24562:      \example_edition\05_Internet');
24563:      \example_edition\06_Communication');
24564:      \example_edition\07_Geographical');
24565:      \example_edition\08_Operating');
24566:      \example_edition\09_Database');
24567:      \example_edition\10_Science');
24568:      \example_edition\11_Embedded');
24569:      \example_edition\12_Security');
24570:
24571: ref:
24572: SHA1:  maXbox3.exe 8B4D7070BA40BDE17EEDBE78121BB8B474A1D6CF
24573: CRC32: maXbox3.exe 1B550873
24574:
24575:   ---- bigbitbox code_cleared_checked----
```