```
1: ********************************************************************************
2:  Constructor Function and Procedure List of maXbox 3.9.9.100
3: ********************************************************************************
4:
5: ////////////////////////////////////////////////////////////////////////////////
6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX
7: --------------------------------------------------------------------------------
8:
9: File Size of EXE: 23070208 V3.9.9.100 Nov 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: ***************Now the Funclist*****************************************
11: Funclist Function : 13556 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 8327  //6792 //6401 4752
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1338 //995 //
16: def head:max: maXbox7: 10.10.2014 09:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: --------------------------------------------------------------------------------
20: Funclist total Size all is: 23221! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 23620
22: ASize of EXE: 23070208 (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.100: B1CFFA2139BEF2D27D17C8212331A1E8C7584BD2
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: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
378: Function Color32( WinColor : TColor) : TColor32;
379: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
380: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale :BOOLean) : TColor
381: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
382: Function ColorToHTML( const Color : TColor) : String
383: function ColorToIdent(Color: Longint; var Ident: string): Boolean)
384: Function ColorToRGB(color: TColor): Longint
385: function ColorToString(Color: TColor): string)
386: Function ColorToWebColorName( Color : TColor) : string
387: Function ColorToWebColorStr( Color : TColor) : string
388: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
389: Function Combination(npr, ncr: integer): extended;
390: Function CombinationInt(npr, ncr: integer): Int64;
391: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
392: Function CommaAdd( const AStr1, AStr2 : String) : string
393: Function CommercialRound( const X : Extended) : Int64
394: Function Commit( grfCommitFlags : Longint) : HResult
395: Function Compare( const NameExt : string) : Boolean
396: function CompareDate(const A, B: TDateTime): TValueRelationship;
397: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer
398: Function CompareFiles(const FN1,FN2 :string; Breathe:TNotifyEvent;BreathingSender:TObject): boolean
399: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
400: Function CompareStr( S1, S2 : string) : Integer
401: function CompareStr(const S1: string; const S2: string): Integer)
402: function CompareString(const S1: string; const S2: string): Integer)
403: Function CompareText( S1, S2 : string) : Integer
404: function CompareText(const S1: string; const S2: string): Integer)
405: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean
406: function CompareTime(const A, B: TDateTime): TValueRelationship;
407: function CompareValueE(const A, B: Extended; Epsilon: Extended = 0): TValueRelationship; overload;
408: function CompareValueD(const A, B: Double; Epsilon: Double = 0): TValueRelationship; overload;
409: function CompareValueS(const A, B: Single; Epsilon: Single = 0): TValueRelationship; overload;
410: function CompareValueI(const A, B: Integer): TValueRelationship; overload;
411: function CompareValueI64(const A, B: Int64): TValueRelationship; overload;
412: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
413: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
414: Function ComponentTypeToString( const ComponentType : DWORD) : string
415: Function ComposeDateTime(Date,Time : TDateTime) : TDateTime;');
416: Function ComponentToStringProc(Component: TComponent): string;
417: Function StringToComponentProc(Value: string): TComponent;
418: Function CompToCurrency( Value : Comp) : Currency
419: Function CompToDouble( Value : Comp) : Double
420: function ComputeFileCRC32(const FileName : String) : Integer;
421: function ComputeSHA256(astr: string; amode: char): string)  //mode F:File, S:String
422: function ComputeSHA512(astr: string; amode: char): string)  //mode F:File, S:String
423: Function Concat(s: string): string
424: Function ConnectAndGetAll : string
425: Function Connected : Boolean
426: function constrain(x, a, b: integer): integer;
427: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
428: Function ConstraintsDisabled : Boolean
429: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
430: Function ContainsState( oState : TniRegularExpressionState) : boolean
431: Function ContainsStr( const AText, ASubText : string) : Boolean
432: Function ContainsText( const AText, ASubText : string) : Boolean
433: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
434: Function Content : string
435: Function ContentFromStream( Stream : TStream) : string
436: Function ContentFromString( const S : string) : string
437: Function CONTROLSDISABLED : BOOLEAN
438: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
439: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
440: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
441: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
442: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
443: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
444: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
445: Function ConvTypeToDescription( const AType : TConvType) : string
```

```
446: Function ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
447: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
448: Function ConvAdd(const AVal:Dbl;const AType1:TConvType;const AVal2:Dbl;const AType2,
     AResultType:TConvType): Double
449: Function ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
     AType2:TConvType): TValueRelationship
450: Function ConvDec( const AValue : Double; const AType, AAmountType : TConvType) : Double;
451: Function ConvDec1(const AValue:Dbl;const AType:TConvType;const AAmount:Dble;const
     AAmountType:TConvType):Double;
452: Function ConvDiff(const AVal1:Dbl;const AType1:TConvType;const AVal2:Dble;const AType2,
     AResuType:TConvType):Double
453: Function ConvInc( const AValue : Double; const AType, AAmountType : TConvType) : Double;
454: Function ConvInc1(const AValue:Dbl;const AType:TConvType;const AAmount:Double;const
     AAmountType:TConvType):Double;
455: Function ConvSameValue(const AValue1:Dbl;const AType1:TConvType;const AValue2:Dbl;const
     AType2:TConvType):Bool
456: Function ConvToStr( const AValue : Double; const AType : TConvType) : string
457: Function ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmount : Double;
     const AAmountType : TConvType) : Boolean
458: Function ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType; const AAmount:Double;const
     AAmountType: TConvType) : Boolean
459: function Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
460: Function CopyFile( Source, Dest : string; CanOverwrite : Boolean) : Boolean
461: Function CopyFileEx( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
462: Function CopyFileTo( const Source, Destination : string) : Boolean
463: function CopyFrom(Source:TStream;Count:Int64):LongInt
464: Function CopyPalette( Palette : HPALETTE) : HPALETTE
465: Function CopyTo( Length : Integer) : string
466: Function CopyTo(stm: IStream; cb: Largeint;out cbRead: Largeint;out cbWritten:Largeint): HResult
467: Function CopyToEOF : string
468: Function CopyToEOL : string
469: Function Cos(e : Extended) : Extended;
470: Function Cosecant( const X : Extended) : Extended
471: Function Cot( const X : Extended) : Extended
472: Function Cotan( const X : Extended) : Extended
473: Function CotH( const X : Extended) : Extended
474: Function Count : Integer
475: Function CountBitsCleared( X : Byte) : Integer;
476: Function CountBitsCleared1( X : Shortint) : Integer;
477: Function CountBitsCleared2( X : Smallint) : Integer;
478: Function CountBitsCleared3( X : Word) : Integer;
479: Function CountBitsCleared4( X : Integer) : Integer;
480: Function CountBitsCleared5( X : Cardinal) : Integer;
481: Function CountBitsCleared6( X : Int64) : Integer;
482: Function CountBitsSet( X : Byte) : Integer;
483: Function CountBitsSet1( X : Word) : Integer;
484: Function CountBitsSet2( X : Smallint) : Integer;
485: Function CountBitsSet3( X : ShortInt) : Integer;
486: Function CountBitsSet4( X : Integer) : Integer;
487: Function CountBitsSet5( X : Cardinal) : Integer;
488: Function CountBitsSet6( X : Int64) : Integer;
489: function countDirfiles(const apath: string): integer;
490: function CountGenerations(Ancestor,Descendent: TClass): Integer
491: Function Coversine( X : Float) : Float
492: function CRC32(const fileName: string): LongWord;
493: Function CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
494: Function CreateColumns : TDBGridColumns
495: Function CreateDataLink : TGridDataLink
496: Function CreateDir( Dir : string) : Boolean
497: function CreateDir(const Dir: string): Boolean)
498: Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
499: Function CreateEnvironmentBlock(const Options:TEnvironmentOptions;const AdditionalVars:TStrings): PChar
500: Function CREATEFIELD(OWNER:TCOMPONENT;PARENTFIELD:TOBJECTFIELD;const
     FIELDNAME:String;CREATECHILDREN:BOOL):TFIELD
501: Function CreateGlobber( sFilespec : string) : TniRegularExpression
502: Function CreateGrayMappedBmp( Handle : HBITMAP) : HBITMAP
503: Function CreateGrayMappedRes( Instance : THandle; ResName : PChar) : HBITMAP
504: function CreateGUID(out Guid: TGUID): HResult)
505: Function CreateInstance( const unkOuter : IUnknown; const iid : TGUID; out obj) : HResult
506: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
507: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
508: Function CreateMessageDialog(const Msg:string; DlgType:TMsgDlgType; Buttons: TMsgDlgButtons) : TForm;
509: Function CreateMessageDialog1(const
     Msg:string;DlgType:TMsgDlgType;Btns:TMsgDlgBtns;DefaultBtn:TMsgDlgBtn):TForm;
510: function CreateOleObject(const ClassName: String): IDispatch;
511: Function CREATEPARAM( FLDTYPE : TFIELDTYPE; const PARAMNAME : String; PARAMTYPE : TPARAMTYPE) : TPARAM
512: Function CreateParameter(const
     Name:WideString;DataType:TDataType;Direction:TParameterDirection;Size:Integer;Value: OleVariant):TParameter
513: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
514: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
515: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
516: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
517: Function CreateValueBuffer( Length : Integer) : TValueBuffer
518: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
519: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
520: Function CreateRotatedFont( Font : TFont; Angle : Integer) : HFONT
521: Function CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
522: Function CreateValueBuffer( Length : Integer) : TValueBuffer
523: Function CreateHexDump( AOwner : TWinControl) : THexDump
```

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

```
612: Function DefaultReadOnly : Boolean
613: Function DefaultWidth : Integer
614: Function DegMinSecToFloat( const Degs, Mins, Secs : Float) : Float
615: Function DegToCycle( const Degrees : Extended) : Extended
616: Function DegToGrad( const Degrees : Extended) : Extended
617: Function DegToGrad( const Value : Extended) : Extended;
618: Function DegToGrad1( const Value : Double) : Double;
619: Function DegToGrad2( const Value : Single) : Single;
620: Function DegToRad( const Degrees : Extended) : Extended
621: Function DegToRad( const Value : Extended) : Extended;
622: Function DegToRad1( const Value : Double) : Double;
623: Function DegToRad2( const Value : Single) : Single;
624: Function DelChar( const pStr : string; const pChar : Char) : string
625: Function DelEnvironmentVar( const Name : string) : Boolean
626: Function Delete( const MsgNum : Integer) : Boolean
627: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
628: Function DeleteFile(const FileName: string): boolean)
629: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
630: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
631: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
632: Function DelSpace( const pStr : string) : string
633: Function DelString( const pStr, pDelStr : string) : string
634: Function DelTree( const Path : string) : Boolean
635: Function Depth : Integer
636: Function Description : string
637: Function DescriptionsAvailable : Boolean
638: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
639: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
640: Function DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
641: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
642: Function DialogsToPixelsX( const Dialogs : Word) : Word
643: Function DialogsToPixelsY( const Dialogs : Word) : Word
644: Function Digits( const X : Cardinal) : Integer
645: Function DirectoryExists( const Name : string) : Boolean
646: Function DirectoryExists( Directory : string) : Boolean
647: Function DiskFree( Drive : Byte) : Int64
648: function DiskFree(Drive: Byte): Int64)
649: Function DiskInDrive( Drive : Char) : Boolean
650: Function DiskSize( Drive : Byte) : Int64
651: function DiskSize(Drive: Byte): Int64)
652: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
653: Function DispatchEnabled : Boolean
654: Function DispatchMask : TMask
655: Function DispatchMethodType : TMethodType
656: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
657: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
658: Function DisplayCase( const S : String) : String
659: Function DisplayRect( Code : TDisplayCode) : TRect
660: Function DisplayRect( TextOnly : Boolean) : TRect
661: Function DisplayStream( Stream : TStream) : string
662: TBufferCoord', 'record Char : integer; Line : integer; end
663: TDisplayCoord', 'record Column : integer; Row : integer; end
664: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord
665: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord
666: Function DomainName( const AHost : String) : String
667: Function DownloadFile(SourceFile, DestFile: string): Boolean;  //fast!
668: Function DownloadFileOpen(SourceFile, DestFile: string): Boolean;  //open process
669: Function DosPathToUnixPath( const Path : string) : string
670: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
671: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
672: Function DoubleToBcd( const AValue : Double) : TBcd;
673: Function DoubleToHex( const D : Double) : string
674: Function DoUpdates : Boolean
675: Function Dragging: Boolean;
676: Function DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
677: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL
678: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
679: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL
680: {Works like InputQuery but displays 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
681: Function DualInputQuery(const ACapt,Prpt1,Prpt2:string;var AVal1,AVal2:string;PasswrdChar:Char= #0):Bool;
682: Function DupeString( const AText : string; ACount : Integer) : string
683: Function Edit : Boolean
684: Function EditCaption : Boolean
685: Function EditText : Boolean
686: Function EditFolderList( Folders : TStrings) : Boolean
687: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
688: Function Elapsed( const Update : Boolean) : Cardinal
689: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
690: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
691: Function EncodeDate( Year, Month, Day : Word) : TDateTime
692: function EncodeDate(Year, Month, Day: Word): TDateTime;
693: Function EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
694: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : TDateTime
695: Function EncodeDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond: Word): TDateTime
696: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
697: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word) : TDateTime
698: Function EncodeString( s : string) : string
699: Function DecodeString( s : string) : string
700: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
```

```
701: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
702: Function EndIP : String
703: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
704: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
705: Function EndOfAMonth( const AYear, AMonth : Word) : TDateTime
706: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
707: Function EndOfAYear( const AYear : Word) : TDateTime
708: Function EndOfTheDay( const AValue : TDateTime) : TDateTime
709: Function EndOfTheMonth( const AValue : TDateTime) : TDateTime
710: Function EndOfTheWeek( const AValue : TDateTime) : TDateTime
711: Function EndOfTheYear( const AValue : TDateTime) : TDateTime
712: Function EndPeriod( const Period : Cardinal) : Boolean
713: Function EndsStr( const ASubText, AText : string) : Boolean
714: Function EndsText( const ASubText, AText : string) : Boolean
715: Function EnsureMsgIDBrackets( const AMsgID : String) : String
716: Function EnsureRange( const AValue, AMin, AMax : Integer) : Integer;
717: Function EnsureRange1( const AValue, AMin, AMax : Int64) : Int64;
718: Function EnsureRange2( const AValue, AMin, AMax : Double) : Double;
719: Function EOF: boolean
720: Function EOln: boolean
721: Function EqualRect( const R1, R2 : TRect) : Boolean
722: function EqualRect(const R1, R2: TRect): Boolean)
723: Function Equals( Strings : TWideStrings) : Boolean
724: function Equals(Strings: TStrings): Boolean;
725: Function EqualState( oState : TniRegularExpressionState) : boolean
726: Function ErrOutput: Text)
727: function ExceptionParam: String;
728: function ExceptionPos: Cardinal;
729: function ExceptionProc: Cardinal;
730: function ExceptionToString(er: TIFException; Param: String): String;
731: function ExceptionType: TIFException;
732: Function ExcludeTrailingBackslash( S : string) : string
733: function ExcludeTrailingBackslash(const S: string): string)
734: Function ExcludeTrailingPathDelimiter( const APath : string) : string
735: Function ExcludeTrailingPathDelimiter( S : string) : string
736: function ExcludeTrailingPathDelimiter(const S: string): string)
737: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): boolean;
738: Function ExecProc : Integer
739: Function ExecSQL : Integer
740: Function ExecSQL( ExecDirect : Boolean) : Integer
741: Function Execute : _Recordset;
742: Function Execute : Boolean
743: Function Execute : Boolean;
744: Function Execute( const SQL : string; Params : TParams; Cache : Boolean; cursor : phDBICur) : Integer
745: Function Execute( const SQL : WideString; Params : TParams; ResultSet : TPSResult) : Integer
746: Function Execute( ParentWnd : HWND) : Boolean
747: Function Execute1(constCommText:WideString;const CType:TCommType;const
     ExecuteOpts:TExecuteOpts):_Recordset;
748: Function Execute1( const Parameters : OleVariant) : _Recordset;
749: Function Execute1( ParentWnd : HWND) : Boolean;
750: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant) : _Recordset;
751: Function ExecuteAction( Action : TBasicAction) : Boolean
752: Function ExecuteDirect( const SQL : WideString) : Integer
753: Function ExecuteFile(const FileName:string;const Params:string;const DefDir:string;ShowCmd:Int):THandle
754: Procedure ExecuteThread2(afunc:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure
755: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle
756: function ExeFileIsRunning(ExeFile: string): boolean;
757: function ExePath: string;
758: function ExePathName: string;
759: Function Exists( AItem : Pointer) : Boolean
760: Function ExitWindows( ExitCode : Cardinal) : Boolean
761: function Exp(x: Extended): Extended;
762: Function ExpandEnvironmentVar( var Value : string) : Boolean
763: Function ExpandFileName( FileName : string) : string
764: function ExpandFileName(const FileName: string): string)
765: Function ExpandUNCFileName( FileName : string) : string
766: function ExpandUNCFileName(const FileName: string): string)
767: Function ExpJ( const X : Float) : Float;
768: Function Exsecans( X : Float) : Float
769: Function Extract( const AByteCount : Integer) : string
770: Function Extract( Item : TClass) : TClass
771: Function Extract( Item : TComponent) : TComponent
772: Function Extract( Item : TObject) : TObject
773: Function ExtractFileDir( FileName : string) : string
774: function ExtractFileDir(const FileName: string): string)
775: Function ExtractFileDrive( FileName : string) : string
776: function ExtractFileDrive(const FileName: string): string)
777: Function ExtractFileExt( FileName : string) : string
778: function ExtractFileExt(const FileName: string): string)
779: Function ExtractFileExtNoDot( const FileName : string) : string
780: Function ExtractFileExtNoDotUpper( const FileName : string) : string
781: Function ExtractFileName( FileName : string) : string
782: function ExtractFileName(const filename: string):string;
783: Function ExtractFilePath( FileName : string) : string
784: function ExtractFilePath(const filename: string):string;
785: Function ExtractRelativePath( BaseName, DestName : string) : string
786: function ExtractRelativePath(const BaseName: string; const DestName: string): string)
787: Function ExtractShortPathName( FileName : string) : string
788: function ExtractShortPathName(const FileName: string): string)
```

```
789: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;Strings: TStrings): Integer
790: function ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings): Integer)
791: Function Fact(numb: integer): Extended;
792: Function FactInt(numb: integer): int64;
793: Function Factorial( const N : Integer) : Extended
794: Function FahrenheitToCelsius( const AValue : Double) : Double
795: function FalseBoolStrs: array of string
796: Function Fetch(var AInput:string;const ADelim:string;const ADelete:Bool;const ACaseSensitive:Bool):string
797: Function FetchCaseInsensitive(var AInput:string; const ADelim:string; const ADelete:Boolean): string
798: Function Fibo(numb: integer): Extended;
799: Function FiboInt(numb: integer): Int64;
800: Function Fibonacci( const N : Integer) : Integer
801: Function FIELDBYNAME( const FIELDNAME : STRING) : TFIELD
802: Function FIELDBYNAME( const FIELDNAME : WIDESTRING) : TFIELD
803: Function FIELDBYNAME( const NAME : String) : TFIELD
804: Function FIELDBYNAME( const NAME : String) : TFIELDDEF
805: Function FIELDBYNUMBER( FIELDNO : INTEGER) : TFIELD
806: Function FileAge( FileName : string) : Integer
807: Function FileAge(const FileName: string): integer)
808: Function FileCompareText( const A, B : String) : Integer
809: Function FileContains(const FileName:string;Text:string;CaseSensitive:Bool;ExceptionOnError:Bool): Boolean
810: Function FileCreate( FileName : string) : Integer;
811: Function FileCreate(const FileName: string): integer)
812: Function FileCreateTemp( var Prefix : string) : THandle
813: Function FileDateToDateTime( FileDate : Integer) : TDateTime
814: function FileDateToDateTime(FileDate: Integer): TDateTime;
815: Function FileExists( const FileName : string) : Boolean
816: Function FileExists( FileName : string) : Boolean
817: function fileExists(const FileName: string): Boolean;
818: Function FileGetAttr( FileName : string) : Integer
819: Function FileGetAttr(const FileName: string): integer)
820: Function FileGetDate( Handle : Integer) : Integer
821: Function FileGetDate(handle: integer): integer
822: Function FileGetDisplayName( const FileName : string) : string
823: Function FileGetSize( const FileName : string) : Integer
824: Function FileGetTempName( const Prefix : string) : string
825: Function FileGetTypeName( const FileName : string) : string
826: Function FileIsReadOnly( FileName : string) : Boolean
827: Function FileLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
828: Function FileOpen( FileName : string; Mode : LongWord) : Integer
829: Function FileOpen(const FileName: string; mode:integer): integer)
830: Function FileRead(handle: integer; Buffer: PChar; count: LongWord): integer
831: Function FileSearch( Name, DirList : string) : string
832: Function FileSearch(const Name, dirList: string): string)
833: Function FileSeek( Handle : Integer; Offset : Int64; Origin : Integer) : Int64;
834: Function FileSeek( Handle, Offset, Origin : Integer) : Integer;
835: Function FileSeek(handle, offset, origin: integer): integer
836: Function FileSetAttr( FileName : string; Attr : Integer) : Integer
837: function FileSetAttr(const FileName: string; Attr: Integer): Integer)
838: Function FileSetDate(FileName : string; Age : Integer) : Integer;
839: Function FileSetDate(handle: integer; age: integer): integer
840: Function FileSetDate2(FileHandle : Integer; Age : Integer) : Integer;
841: Function FileSetDateH( Handle : Integer; Age : Integer) : Integer;
842: Function FileSetReadOnly( FileName : string; ReadOnly : Boolean) : Boolean
843: Function FileSize( const FileName : string) : int64
844: Function FileSizeByName( const AFilename : string) : Longint
845: function FileWrite(Handle: Integer; const Buffer: pChar; Count: LongWord): Integer)
846: Function FilterSpecArray : TComdlgFilterSpecArray
847: Function FIND( ACAPTION : String) : TMENUITEM
848: Function Find( AItem : Pointer; out AData : Pointer) : Boolean
849: Function FIND( const ANAME : String) : TNAMEDITEM
850: Function Find( const DisplayName : string) : TAggregate
851: Function Find( const Item : TBookmarkStr; var Index : Integer) : Boolean
852: Function FIND( const NAME : String) : TFIELD
853: Function FIND( const NAME : String) : TFIELDDEF
854: Function FIND( const NAME : String) : TINDEXDEF
855: Function Find( const S : WideString; var Index : Integer) : Boolean
856: function Find(S:String;var Index:Integer):Boolean
857: Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
858: Function FindBand( AControl : TControl) : TCoolBand
859: Function FindBoundary( AContentType : string) : string
860: Function FindButton( AModalResult : TModalResult) : TTaskDialogBaseButtonItem
861: Function FindCaption(StartIndex: Integer;Value: string; Partial,Inclusive,Wrap: Boolean): TListItem
862: Function FindCdLineSwitch( Switch : string; IgnoreCase : Boolean) : Boolean;
863: Function FindCloseW(FindFile: Integer): LongBool; stdcall;
864: Function FindCmdLineSwitch( Switch : string; Chars : TSysCharSet; IgnoreCase : Boolean) : Boolean;
865: Function FindCmmdLineSwitch( Switch : string) : Boolean;
866: function FindComponent(AName: String): TComponent;
867: function FindComponent(vlabel: string): TComponent;
868: function FindComponent2(vlabel: string): TComponent;
869: function FindControl(Handle: HWnd): TWinControl;
870: Function FindData( StartIndex : Integer; Value : Pointer; Inclusive, Wrap : Boolean) : TListItem
871: Function FindDatabase( const DatabaseName : string) : TDatabase
872: function FindDragTarget(const Pos: TPoint; AllowDisabled: Boolean): TControl;
873: Function FINDFIELD( const FIELDNAME : STRING) : TFIELD
874: Function FINDFIELD( const FIELDNAME : WideString) : TFIELD
875: Function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec):Integer)
876: Function FindNext2(var F: TSearchRec): Integer)
877: procedure FindClose2(var F: TSearchRec)
```

```
878: Function FINDFIRST : BOOLEAN
879: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
880:   sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms,sfRecent,sfSendTo,
     sfStartMenu, stStartUp, sfTemplates);
881:  FFolder: array [TJvSpecialFolder] of Integer =
882:     (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
883:      CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
884:      CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
885:      CSIDL_STARTUP, CSIDL_TEMPLATES);
886: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean);
887: function Findfirst(const filepath: string; attr: integer): integer;
888: function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer)
889: Function FindFirstNotOf( AFind, AText : String) : Integer
890: Function FindFirstOf( AFind, AText : String) : Integer
891: Function FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
892: Function FINDINDEXFORFIELDS( const FIELDS : String) : TINDEXDEF
893: Function FindInstanceOf( AClass : TClass; AExact : Boolean; AStartAt : Integer) : Integer
894: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND) : TMENUITEM
895: function FindItemId( Id : Integer) : TCollectionItem
896: Function FindKey( const KeyValues : array of const) : Boolean
897: Function FINDLAST : BOOLEAN
898: Function FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
899: Function FindModuleClass( AClass : TComponentClass) : TComponent
900: Function FindModuleName( const AClass : string) : TComponent
901: Function FINDNEXT : BOOLEAN
902: function FindNext: integer;
903: function FindNext2(var F: TSearchRec): Integer)
904: Function FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible : Boolean) : TTabSheet
905: Function FindNextToSelect : TTreeNode
906: Function FINDPARAM( const VALUE : String) : TPARAM
907: Function FindParam( const Value : WideString) : TParameter
908: Function FINDPRIOR : BOOLEAN
909: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar) : TResourceHandle
910: Function FindSession( const SessionName : string) : TSession
911: function FindStringResource(Ident: Integer): string)
912: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer
913: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString) : AnsiString
914: function FindVCLWindow(const Pos: TPoint): TWinControl;
915: function FindWindow(C1, C2: PChar): Longint;
916: Function FindInPaths(const fileName,paths: String): String;
917: Function Finger : String
918: Function First : TClass
919: Function First : TComponent
920: Function First : TObject
921: Function FirstDelimiter( const delimiters : string; const Str : String) : integer;
922: Function FirstDelimiter1( const delimiters : WideString; const Str : WideString) : integer;
923: Function FirstInstance( const ATitle : string) : Boolean
924: Function FloatPoint( const X, Y : Float) : TFloatPoint;
925: Function FloatPoint1( const P : TPoint) : TFloatPoint;
926: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) : Boolean
927: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
928: Function FloatRect1( const Rect : TRect) : TFloatRect;
929: Function FloatsEqual( const X, Y : Float) : Boolean
930: Function FloatToBin(const D: Double): string;     //doubletohex -> hextobin! in buffer
931: Function FloatToCurr( Value : Extended) : Currency
932: Function FloatToDateTime( Value : Extended) : TDateTime
933: Function FloatToStr( Value : Extended) : string;
934: Function FloatToStr(e : Extended) : String;
935: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer) : string;
936: function FloatToStrF(Value: Extended; Format: TFloatFormat;Precision:Integer; Digits: Integer): string)
937: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings
     : TFormatSettings) : string;
938: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer;
     FormatSettings : TFormatSettings) : string;
939: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue;Format: TFloatFormat;
     Precision,Digits: Integer): Integer)
940: Function Floor( const X : Extended) : Integer
941: Function FloorInt( Value : Integer; StepSize : Integer) : Integer
942: Function FloorJ( const X : Extended) : Integer
943: Function Flush( const Count : Cardinal) : Boolean
944: Function Flush(var t: Text): Integer
945: function FmtLoadStr(Ident: Integer; const Args: array of const): string)
946: function FOCUSED:BOOLEAN
947: Function ForceBackslash( const PathName : string) : string
948: Function ForceDirectories( const Dir : string) : Boolean
949: Function ForceDirectories( Dir : string) : Boolean
950: Function ForceDirectories( Name : string) : Boolean
951: Function ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
952: Function ForceInRange( A, Min, Max : Integer) : Integer
953: Function ForceInRangeR( const A, Min, Max : Double) : Double
954: Function ForEach( AProc : TBucketProc; AInfo : Pointer) : Boolean;
955: Function ForEach1( AEvent : TBucketEvent) : Boolean;
956: Function ForegroundTask: Boolean
957: function Format(const Format: string; const Args: array of const): string;
958: Function FormatBcd( const Format : string; Bcd : TBcd) : string
959: FUNCTION FormatBigInt(s: string): STRING;
960: function FormatByteSize(const bytes: int64): string;
961: function FormatBuf(var Buffer:PChar;BufLen:Card;const Format:string;FmtLen:Card;const Args:array of
     const):Cardinal
```

```
 962: Function FormatCount( iCount : integer; const sSingular : string; const sPlural : string) : string
 963: Function FormatCurr( Format : string; Value : Currency) : string;
 964: function FormatCurr(const Format: string; Value: Currency): string)
 965: Function FormatDateTime( Format : string; DateTime : TDateTime) : string;
 966: function FormatDateTime(const fmt: string; D: TDateTime): string;
 967: Function FormatFloat( Format : string; Value : Extended) : string;
 968: function FormatFloat(const Format: string; Value: Extended): string)
 969: Function FormatFloat( Format : string; Value : Extended) : string;
 970: Function FormatFloat2( Format : string; Value : Extended; FormatSettings : TFormatSettings) : string;
 971: Function FormatCurr( Format : string; Value : Currency) : string;
 972: Function FormatCurr2(Format: string; Value : Currency; FormatSettings : TFormatSettings) : string;
 973: Function Format2(const Format:string;const Args:array of const;const FSettings:TFormatSettings): string
 974: FUNCTION FormatInt(i: integer): STRING;
 975: FUNCTION FormatInt64(i: int64): STRING;
 976: Function FormatMaskText( const EditMask : string; const Value : string) : string
 977: Function FormatValue( AValue : Cardinal) : string
 978: Function FormatVersionString( const HiV, LoV : Word) : string;
 979: Function FormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
 980: function Frac(X: Extended): Extended);
 981: Function FreeResource( ResData : HGLOBAL) : LongBool
 982: Function FromCommon( const AValue : Double) : Double
 983: function FromCommon(const AValue: Double): Double;
 984: Function FTPGMTDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 985: Function FTPLocalDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 986: Function FTPMLSToGMTDateTime( const ATimeStamp : String) : TDateTime
 987: Function FTPMLSToLocalDateTime( const ATimeStamp : String) : TDateTime
 988: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
 989: //Function Funclist Size is: 6444 of mX3.9.8.9
 990: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:
      TPaymentTime):Extended
 991: Function FullTimeToStr(SUMTime: TDateTime): string;');
 992: Function Gauss( const x, Spread : Double) : Double
 993: function Gauss(const x,Spread: Double): Double;
 994: Function GCD(x, y : LongInt) : LongInt;
 995: Function GCDJ( X, Y : Cardinal) : Cardinal
 996: Function GDAL: LongWord
 997: Function GdiFlush : BOOL
 998: Function GdiSetBatchLimit( Limit : DWORD) : DWORD
 999: Function GdiGetBatchLimit : DWORD
1000: Function GenerateHeader : TIdHeaderList
1001: Function GeometricMean( const X : TDynFloatArray) : Float
1002: Function Get( AURL : string) : string;
1003: Function Get2( AURL : string) : string;
1004: Function Get8087CW : Word
1005: function GetActiveOleObject(const ClassName: String): IDispatch;
1006: Function GetAliasDriverName( const AliasName : string) : string
1007: Function GetAPMBatteryFlag : TAPMBatteryFlag
1008: Function GetAPMBatteryFullLifeTime : DWORD
1009: Function GetAPMBatteryLifePercent : Integer
1010: Function GetAPMBatteryLifeTime : DWORD
1011: Function GetAPMLineStatus : TAPMLineStatus
1012: Function GetAppdataFolder : string
1013: Function GetAppDispatcher : TComponent
1014: function GetArrayLength: integer;
1015: Function GetASCII: string;
1016: Function GetASCIILine: string;
1017: Function GetAsHandle( Format : Word) : THandle
1018: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1019: Function GetBackupFileName( const FileName : string) : string
1020: function GetBaseAddress(PID:DWORD):DWORD;  //Process API
1021: Function GetBBitmap( Value : TBitmap) : TBitmap
1022: Function GetBIOSCopyright : string
1023: Function GetBIOSDate : TDateTime
1024: Function GetBIOSExtendedInfo : string
1025: Function GetBIOSName : string
1026: Function getBitmap(apath: string): TBitmap;
1027: Function GetBitmap( Index : Integer; Image : TBitmap) : Boolean  //object
1028: Function getBitMapObject(const bitmappath: string): TBitmap;
1029: Function GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1030: Function GetCapsLockKeyState : Boolean
1031: function GetCaptureControl: TControl;
1032: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char) : TJclCdTrackInfo;
1033: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char) : string;
1034: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) : string
1035: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char) : string
1036: Function GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1037: Function GetClockValue : Int64
1038: function getCmdLine: PChar;
1039: function getCmdShow: Integer;
1040: function GetCPUSpeed: Double;
1041: Function GetColField( DataCol : Integer) : TField
1042: Function GetColorBlue( const Color : TColor) : Byte
1043: Function GetColorFlag( const Color : TColor) : Byte
1044: Function GetColorGreen( const Color : TColor) : Byte
1045: Function GetColorRed( const Color : TColor) : Byte
1046: Function GetComCtlVersion : Integer
1047: Function GetComPorts: TStringlist;
1048: Function GetCommonAppdataFolder : string
1049: Function GetCommonDesktopdirectoryFolder : string
```

```
1050: Function GetCommonFavoritesFolder : string
1051: Function GetCommonFilesFolder : string
1052: Function GetCommonProgramsFolder : string
1053: Function GetCommonStartmenuFolder : string
1054: Function GetCommonStartupFolder : string
1055: Function GetComponent( Owner, Parent : TComponent) : TComponent
1056: Function GetConnectionRegistryFile( DesignMode : Boolean) : string
1057: Function GetCookiesFolder : string
1058: Function GetCPUSpeed( var CpuSpeed : TFreqInfo) : Boolean
1059: Function GetCurrent : TFavoriteLinkItem
1060: Function GetCurrent : TListItem
1061: Function GetCurrent : TTaskDialogBaseButtonItem
1062: Function GetCurrent : TToolButton
1063: Function GetCurrent : TTreeNode
1064: Function GetCurrent : WideString
1065: Function GetCurrentDir : string
1066: function GetCurrentDir: string)
1067: Function GetCurrentFolder : string
1068: Function GETCURRENTRECORD( BUFFER : PCHAR) : BOOLEAN
1069: Function GetCurrentProcessId : TIdPID
1070: Function GetCurrentThreadHandle : THandle
1071: Function GetCurrentThreadID: LongWord; stdcall;
1072: Function GetCustomHeader( const Name : string) : String
1073: Function GetDataItem( Value : Pointer) : Longint
1074: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string) : Integer;
1075: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string) : Integer;
1076: Function GETDATASIZE : INTEGER
1077: Function GetDC(hdwnd: HWND): HDC;
1078: Function GetDefaultFileExt( const MIMEType : string) : string
1079: Function GetDefaults : Boolean
1080: Function GetDefaultSchemaName : WideString
1081: Function GetDefaultStreamLoader : IStreamLoader
1082: Function GetDesktopDirectoryFolder : string
1083: Function GetDesktopFolder : string
1084: Function GetDFAState( oStates : TList) : TniRegularExpressionState
1085: Function GetDirectorySize( const Path : string) : Int64
1086: Function GetDisplayWidth : Integer
1087: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer) : Boolean
1088: Function GetDomainName : string
1089: Function GetDriverRegistryFile( DesignMode : Boolean) : string
1090: function GetDriveType(rootpath: pchar): cardinal;
1091: Function GetDriveTypeStr( const Drive : Char) : string
1092: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1093: Function GetEnumerator : TListItemsEnumerator
1094: Function GetEnumerator : TTaskDialogButtonsEnumerator
1095: Function GetEnumerator : TToolBarEnumerator
1096: Function GetEnumerator : TTreeNodesEnumerator
1097: Function GetEnumerator : TWideStringsEnumerator
1098: Function GetEnvVar( const VarName : string) : string
1099: Function GetEnvironmentVar( const AVariableName : string) : string
1100: Function GetEnvironmentVariable( const VarName : string) : string
1101: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean) : Boolean
1102: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean) : Boolean
1103: Function getEnvironmentString: string;
1104: Function GetExceptionHandler : TObject
1105: Function GetFavoritesFolder : string
1106: Function GetFieldByName( const Name : string) : string
1107: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo) : Boolean
1108: Function GetFieldValue( ACol : Integer) : string
1109: Function GetFileAgeCoherence( const FileName : string) : Boolean
1110: Function GetFileCreation( const FileName : string) : TFileTime
1111: Function GetFileCreationTime( const Filename : string) : TDateTime
1112: Function GetFileInformation( const FileName : string) : TSearchRec
1113: Function GetFileLastAccess( const FileName : string) : TFileTime
1114: Function GetFileLastWrite( const FileName : string) : TFileTime
1115: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1116: Function GetFileList1(apath: string): TStringlist;
1117: Function GetFileMIMEType( const AFileName : string) : string
1118: Function GetFileSize( const FileName : string) : Int64
1119: Function GetFileVersion( AFileName : string) : Cardinal
1120: Function GetFileVersion( const AFilename : string) : Cardinal
1121: Function GetFileSize2(Handle: Integer; x: Integer): Integer; stdcall;
1122: Function GetFileDate(aFile:string; aWithTime:Boolean):string;
1123: Function GetFilterData( Root : PExprNode) : TExprData
1124: Function getFirstChild : LongInt
1125: Function getFirstChild : TTreeNode
1126: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string) : string
1127: Function GetFirstNode : TTreeNode
1128: Function GetFontsFolder : string
1129: Function GetFormulaValue( const Formula : string) : Extended
1130: Function GetFreePageFileMemory : Integer
1131: Function GetFreePhysicalMemory : Integer
1132: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind) : Integer;
1133: Function GetFreeSystemResources1 : TFreeSystemResources;
1134: Function GetFreeVirtualMemory : Integer
1135: Function GetFromClipboard : Boolean
1136: Function GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet) : String
1137: Function GetGBitmap( Value : TBitmap) : TBitmap
1138: Function GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
```

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

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

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

```
1400: Function GotoKey : Boolean
1401: Function GradToCycle( const Grads : Extended) : Extended
1402: Function GradToDeg( const Grads : Extended) : Extended
1403: Function GradToDeg( const Value : Extended) : Extended;
1404: Function GradToDeg1( const Value : Double) : Double;
1405: Function GradToDeg2( const Value : Single) : Single;
1406: Function GradToRad( const Grads : Extended) : Extended
1407: Function GradToRad( const Value : Extended) : Extended;
1408: Function GradToRad1( const Value : Double) : Double;
1409: Function GradToRad2( const Value : Single) : Single;
1410: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1411: Function GreenComponent( const Color32 : TColor32) : Integer
1412: function GUIDToString(const GUID: TGUID): string)
1413: Function HandleAllocated : Boolean
1414: function HandleAllocated: Boolean;
1415: Function HandleRequest : Boolean
1416: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1417: Function HarmonicMean( const X : TDynFloatArray) : Float
1418: Function HasAsParent( Value : TTreeNode) : Boolean
1419: Function HASCHILDDEFS : BOOLEAN
1420: Function HasCurValues : Boolean
1421: Function HasExtendCharacter( const s : UTF8String) : Boolean
1422: Function HasFormat( Format : Word) : Boolean
1423: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1424: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1425: Function HashValue(AStream: TStream): LongWord
1426: Function HashValue(AStream: TStream): Word
1427: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1428: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1429: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1430: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1431: Function HashValue16( const ASrc : string) : Word;
1432: Function HashValue16stream( AStream : TStream) : Word;
1433: Function HashValue32( const ASrc : string) : LongWord;
1434: Function HashValue32Stream( AStream : TStream) : LongWord;
1435: Function HasMergeConflicts : Boolean
1436: Function hasMoreTokens : boolean
1437: Function HASPARENT : BOOLEAN
1438: function HasParent: Boolean
1439: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1440: Function HasUTF8BOM( S : TStream) : boolean;
1441: Function HasUTF8BOM1( S : AnsiString) : boolean;
1442: Function Haversine( X : Float) : Float
1443: Function Head( s : string; const subs : string; var tail : string) : string
1444: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1445: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1446: function HELPJUMP(JUMPID:STRING):BOOLEAN
1447: Function HeronianMean( const a, b : Float) : Float
1448: function HexStrToStr(Value: string): string;
1449: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1450: function HexToBin2(HexNum: string): string;
1451: Function HexToDouble( const Hex : string) : Double
1452: function HexToInt(hexnum: string): LongInt;
1453: function HexToStr(Value: string): string;
1454: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1455: function Hi(vdat: word): byte;
1456: function HiByte(W: Word): Byte)
1457: function High: Int64;
1458: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1459: function HINSTANCE: longword;
1460: function HiWord(l: DWORD): Word)
1461: function HMODULE: longword;
1462: Function HourOf( const AValue : TDateTime) : Word
1463: Function HourOfTheDay( const AValue : TDateTime) : Word
1464: Function HourOfTheMonth( const AValue : TDateTime) : Word
1465: Function HourOfTheWeek( const AValue : TDateTime) : Word
1466: Function HourOfTheYear( const AValue : TDateTime) : Word
1467: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1468: Function HourSpan( const ANow, AThen : TDateTime) : Double
1469: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1470: Function HTMLDecode( const AStr : String) : String
1471: Function HTMLEncode( const AStr : String) : String
1472: Function HTMLEscape( const Str : string) : string
1473: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1474: Function HTTPDecode( const AStr : String) : string
1475: Function HTTPEncode( const AStr : String) : string
1476: Function Hypot( const X, Y : Extended) : Extended
1477: Function IBMax( n1, n2 : Integer) : Integer
1478: Function IBMin( n1, n2 : Integer) : Integer
1479: Function IBRandomString( iLength : Integer) : String
1480: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1481: Function IBStripString( st : String; CharsToStrip : String) : String
1482: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1483: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1484: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1485: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1486: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1487: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1488:  Function RandomString( iLength : Integer) : String');
```

```
1489:  Function RandomInteger( iLow, iHigh : Integer) : Integer');
1490:  Function StripString( st : String; CharsToStrip : String) : String');
1491:  Function FormatIdentifier( Dialect : Integer; Value : String) : String');
1492:  Function FormatIdentifierValue( Dialect : Integer; Value : String) : String');
1493:  Function ExtractIdentifier( Dialect : Integer; Value : String) : String');
1494:  Function QuoteIdentifier( Dialect : Integer; Value : String) : String');
1495:  Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
1496:  Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String;
1497:  function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1498: Function IconToBitmap( Ico : HICON) : TBitmap
1499: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1500: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1501: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean)
1502: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1503: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1504: Function IdGetDefaultCharSet : TIdCharSet
1505: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1506: Function IdPorts2 : TStringList
1507: Function IdToMib( const Id : string) : string
1508: Function IdSHA1Hash(apath: string): string;
1509: Function IdHashSHA1(apath: string): string;
1510: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1511: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1512: Function IfThenInt( AValue : Boolean; const ATrue : integer; AFalse : integer): integer;');
1513: Function IfThenDouble( AValue : Boolean; const ATrue : double; AFalse : double): double;');
1514: Function IfThenBool( AValue : Boolean; const ATrue : boolean; AFalse : boolean): boolean;');
1515: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1516: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1517: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
1518: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1519: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1520: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1521: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1522: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1523: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1524: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1525: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1526: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1527: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1528: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1529: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1530: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1531: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1532: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1533: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1534: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1535: Function IncludeTrailingBackslash( S : string) : string
1536: function IncludeTrailingBackslash(const S: string): string)
1537: Function IncludeTrailingPathDelimiter( const APath : string) : string
1538: Function IncludeTrailingPathDelimiter( S : string) : string
1539: function IncludeTrailingPathDelimiter(const S: string): string)
1540: Function IncludeTrailingSlash( const APath : string) : string
1541: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliSeconds : Int64) : TDateTime
1542: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1543: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1544: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1545: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1546: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1547: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1548: Function IndexOf( AClass : TClass) : Integer
1549: Function IndexOf( AComponent : TComponent) : Integer
1550: Function IndexOf( AObject : TObject) : Integer
1551: Function INDEXOF( const ANAME : String) : INTEGER
1552: Function IndexOf( const DisplayName : string) : Integer
1553: Function IndexOf( const Item : TBookmarkStr) : Integer
1554: Function IndexOf( const S : WideString) : Integer
1555: Function IndexOf( const View : TJclFileMappingView) : Integer
1556: Function INDEXOF( FIELD : TFIELD) : INTEGER
1557: Function IndexOf( ID : LCID) : Integer
1558: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1559: Function IndexOf( Value : TListItem) : Integer
1560: Function IndexOf( Value : TTreeNode) : Integer
1561: function IndexOf(const S: string): Integer;
1562: Function IndexOfName( const Name : WideString) : Integer
1563: function IndexOfName(Name: string): Integer;
1564: Function IndexOfObject( AObject : TObject) : Integer
1565: function IndexofObject(AObject:tObject):Integer
1566: Function IndexOfTabAt( X, Y : Integer) : Integer
1567: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1568: Function IndexText( const AText : string; const AValues : array of string) : Integer
1569: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer
1570: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer
1571: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer
1572: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1573: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1574: Function IndyGetHostName : string
1575: Function IndyInterlockedDecrement( var I : Integer) : Integer
1576: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1577: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
```

```
1578: Function IndyInterlockedIncrement( var I : Integer) : Integer
1579: Function IndyLowerCase( const A1 : string) : string
1580: Function IndyStrToBool( const AString : String) : Boolean
1581: Function IndyUpperCase( const A1 : string) : string
1582: Function InitCommonControl( CC : Integer) : Boolean
1583: Function InitTempPath : string
1584: Function InMainThread : boolean
1585: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1586: Function Input: Text)
1587: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1588: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1589: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1590: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1591: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1592: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1593: Function InRangeR( const A, Min, Max : Double) : Boolean
1594: function Insert( Index : Integer) : TCollectionItem
1595: Function Insert( Index : Integer) : TComboExItem
1596: Function Insert( Index : Integer) : THeaderSection
1597: Function Insert( Index : Integer) : TListItem
1598: Function Insert( Index : Integer) : TStatusPanel
1599: Function Insert( Index : Integer) : TWorkArea
1600: Function Insert( Index : LongInt; const Text : string) : LongInt
1601: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1602: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1603: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1604: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1605: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1606: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1607: Function Instance : Longint
1608: function InstanceSize: Longint
1609: Function Int(e : Extended) : Extended;
1610: function Int64ToStr(i: Int64): String;
1611: Function IntegerToBcd( const AValue : Integer) : TBcd
1612: Function Intensity( const Color32 : TColor32) : Integer;
1613: Function Intensity( const R, G, B : Single) : Single;
1614: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
      FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1615: Function InterestRate(NPeriods:Integer;const Payment,PresVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1616: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1617: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1618: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1619: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1620: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1621: Function IntMibToStr( const Value : string) : string
1622: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1623: Function IntToBin( Value : cardinal) : string
1624: Function IntToHex( Value : Integer; Digits : Integer) : string;
1625: function IntToHex(a: integer; b: integer): string;
1626: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1627: function IntToHex64(Value: Int64; Digits: Integer): string)
1628: Function IntTo3Str( Value : Longint; separator: string) : string
1629: Function inttobool( aInt : LongInt) : Boolean
1630: function IntToStr(i: Int64): String;
1631: Function IntToStr64(Value: Int64): string)
1632: function IOResult: Integer
1633: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1634: Function IsAccel(VK: Word; const Str: string): Boolean
1635: Function IsAddressInNetwork( Address : String) : Boolean
1636: Function IsAdministrator : Boolean
1637: Function IsAlias( const Name : string) : Boolean
1638: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1639: Function IsASCII( const AByte : Byte) : Boolean;
1640: Function IsASCIILDH( const AByte : Byte) : Boolean;
1641: Function IsAssembly(const FileName: string): Boolean;
1642: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1643: Function IsBinary(const AChar : Char) : Boolean
1644: function IsConsole: Boolean)
1645: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1646: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1647: Function IsDelphiDesignMode : boolean
1648: Function IsDelphiRunning : boolean
1649: Function IsDFAState : boolean
1650: Function IsDirectory( const FileName : string) : Boolean
1651: Function IsDomain( const S : String) : Boolean
1652: function IsDragObject(Sender: TObject): Boolean;
1653: Function IsEditing : Boolean
1654: Function ISEMPTY : BOOLEAN
1655: Function IsEqual( Value : TParameters) : Boolean
1656: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1657: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1658: Function IsFirstNode : Boolean
1659: Function IsFloatZero( const X : Float) : Boolean
1660: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1661: Function IsFormOpen(const FormName: string): Boolean;
1662: Function IsFQDN( const S : String) : Boolean
1663: Function IsGrayScale : Boolean
1664: Function IsHex( AChar : Char) : Boolean;
```

```
1665: Function IsHexString(const AString: string): Boolean;
1666: Function IsHostname( const S : String) : Boolean
1667: Function IsInfinite( const AValue : Double) : Boolean
1668: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1669: Function IsInternet: boolean;
1670: Function IsLeadChar( ACh : Char) : Boolean
1671: Function IsLeapYear( Year : Word) : Boolean
1672: function IsLeapYear(Year: Word): Boolean)
1673: function IsLibrary: Boolean)
1674: Function ISLINE : BOOLEAN
1675: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1676: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1677: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1678: Function IsMatch( const Pattern, Text : string) : Boolean  //Grep like RegEx
1679: Function IsMainAppWindow( Wnd : HWND) : Boolean
1680: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1681: function IsMemoryManagerSet: Boolean)
1682: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1683: function IsMultiThread: Boolean)
1684: Function IsNumeric( AChar : Char) : Boolean;
1685: Function IsNumeric2( const AString : string) : Boolean;
1686: Function IsNTFS: Boolean;
1687: Function IsOctal( AChar : Char) : Boolean;
1688: Function IsOctalString(const AString: string) : Boolean;
1689: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1690: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1691: Function IsPM( const AValue : TDateTime) : Boolean
1692: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1693: Function IsPortAvailable( ComNum : Cardinal) : Boolean');
1694: Function IsCOMPortReal( ComNum : Cardinal) : Boolean');
1695: Function IsCOM( ComNum : Cardinal) : Boolean');
1696: Function IsCOMPort: Boolean');
1697: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1698: Function IsPrimeRM( N : Cardinal) : Boolean  //rabin miller
1699: Function IsPrimeTD( N : Cardinal) : Boolean  //trial division
1700: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1701: Function ISqrt( const I : Smallint) : Smallint
1702: Function IsReadOnly(const Filename: string): boolean;
1703: Function IsRectEmpty( const Rect : TRect) : Boolean
1704: function IsRectEmpty(const Rect: TRect): Boolean)
1705: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1706: Function ISRIGHTTOLEFT : BOOLEAN
1707: function IsRightToLeft: Boolean
1708: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1709: Function ISSEQUENCED : BOOLEAN
1710: Function IsSystemModule( const Module : HMODULE) : Boolean
1711: Function IsSystemResourcesMeterPresent : Boolean
1712: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1713: Function IsToday( const AValue : TDateTime) : Boolean
1714: function IsToday(const AValue: TDateTime): Boolean;
1715: Function IsTopDomain( const AStr : string) : Boolean
1716: Function IsUTF8LeadByte( Lead : Char) : Boolean
1717: Function IsUTF8String( const s : UTF8String) : Boolean
1718: Function IsUTF8TrailByte( Lead : Char) : Boolean
1719: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1720: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1721: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1722: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1723: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1724: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1725: Function IsValidIdent( Ident : string) : Boolean
1726: function IsValidIdent1(const Ident: string; AllowDots: Boolean): Boolean)
1727: Function IsValidIP( const S : String) : Boolean
1728: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1729: Function IsValidPNG(stream: TStream): Boolean;
1730: Function IsValidJPEG(stream: TStream): Boolean;
1731: Function IsValidISBN( const ISBN : AnsiString) : Boolean
1732: Function IsVariantManagerSet: Boolean; //deprecated;
1733: Function IsVirtualPcGuest : Boolean;
1734: Function IsVmWareGuest : Boolean;
1735: Function IsVCLControl(Handle: HWnd): Boolean;
1736: Function IsWhiteString( const AStr : String) : Boolean
1737: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1738: Function IsWoW64: boolean;
1739: Function IsWin64: boolean;
1740: Function IsWow64String(var s: string): Boolean;
1741: Function IsWin64String(var s: string): Boolean;
1742: Function IsWindowsVista: boolean;
1743: Function isPowerof2(num: int64): boolean;
1744: Function powerOf2(exponent: integer): int64;
1745: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1746: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1747: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1748: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1749: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1750: Function ItemRect( Index : Integer) : TRect
1751: function ITEMRECT(INDEX:INTEGER):TRECT
1752: Function ItemWidth( Index : Integer) : Integer
1753: Function JavahashCode(val: string): Integer;
```

```
1754: Function JosephusG(n,k: integer; var graphout: string): integer;
1755: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1756: Function JustName(PathName : string) : string;  //in path and ext
1757: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1758: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1759: Function KeepAlive : Boolean
1760: Function KeysToShiftState(Keys: Word): TShiftState;
1761: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1762: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1763: Function KeyboardStateToShiftState: TShiftState; overload;
1764: Function Languages : TLanguages
1765: Function Last : TClass
1766: Function Last : TComponent
1767: Function Last : TObject
1768: Function LastDelimiter( Delimiters, S : string) : Integer
1769: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1770: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1771: Function Latitude2WGS84(lat: double): double;
1772: Function LCM(m,n:longint):longint;
1773: Function LCMJ( const X, Y : Cardinal) : Cardinal
1774: Function Ldexp( const X : Extended; const P : Integer) : Extended
1775: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1776: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1777: function Length: Integer;
1778: Procedure LetFileList(FileList: TStringlist; apath: string);
1779: function lengthmp3(mp3path: string):integer;
1780: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1781: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1782: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1783: function LineStart(Buffer, BufPos: PChar): PChar
1784: function LineStart(Buffer, BufPos: PChar): PChar)
1785: function ListSeparator: char;
1786: function Ln(x: Extended): Extended;
1787: Function LnXP1( const X : Extended) : Extended
1788: function Lo(vdat: word): byte;
1789: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1790: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1791: Function LoadFileAsString( const FileName : string) : string
1792: Function LoadFromFile( const FileName : string) : TBitmapLoader
1793: function LoadFile(const FileName: TFileName): string;
1794: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1795: Function LoadPackage(const Name: string): HMODULE
1796: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1797: Function LoadStr( Ident : Integer) : string
1798: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1799: Function LoadWideStr( Ident : Integer) : WideString
1800: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1801: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1802: Function LockServer( fLock : LongBool) : HResult
1803: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1804: Function Log( const X : Extended) : Extended
1805: Function Log10( const X : Extended) : Extended
1806: Function Log2( const X : Extended) : Extended
1807: function LogBase10(X: Float): Float;
1808: Function LogBase2(X: Float): Float;
1809: Function LogBaseN(Base, X: Float): Float;
1810: Function LogN( const Base, X : Extended) : Extended
1811: Function LogOffOS : Boolean
1812: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1813: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1814: Function LongDateFormat: string;
1815: function LongTimeFormat: string;
1816: Function LongWordToFourChar( ACardinal : LongWord) : string
1817: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1818: Function LookupName( const name : string) : TInAddr
1819: Function LookupService( const service : string) : Integer
1820: function Low: Int64;
1821: Function LowerCase( S : string) : string
1822: Function Lowercase(s : AnyString) : AnyString;
1823: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1824: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1825: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1826: function MainInstance: longword
1827: function MainThreadID: longword
1828: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1829: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1830: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1831: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1832: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1833: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1834: Function MakeFile(const FileName: string): integer)');
1835: function MakeLong(A, B: Word): Longint)
1836: Function MakeTempFilename( const APath : String) : string
1837: Function MakeValidFileName( const Str : string) : string
1838: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1839: function MakeWord(A, B: Byte): Word)
1840: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1841: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
```

```
1842: Function MapValues( Mapping : string; Value : string) : string
1843: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1844: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1845: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1846: Function MaskGetFldSeparator( const EditMask : string) : Integer
1847: Function MaskGetMaskBlank( const EditMask : string) : Char
1848: Function MaskGetMaskSave( const EditMask : string) : Boolean
1849: Function MaskIntlLiteralToChar( IChar : Char) : Char
1850: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1851: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1852: Function MaskString( Mask, Value : String) : String
1853: Function Match( const sString : string) : TniRegularExpressionMatchResul
1854: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1855: Function Matches( const Filename : string) : Boolean
1856: Function MatchesMask( const Filename, Mask : string) : Boolean
1857: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1858: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1859: Function Max( AValueOne, AValueTwo : Integer) : Integer
1860: function Max(const x,y: Integer): Integer;
1861: Function Max1( const B1, B2 : Shortint) : Shortint;
1862: Function Max2( const B1, B2 : Smallint) : Smallint;
1863: Function Max3( const B1, B2 : Word) : Word;
1864: function Max3(const x,y,z: Integer): Integer;
1865: Function Max4( const B1, B2 : Integer) : Integer;
1866: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1867: Function Max6( const B1, B2 : Int64) : Int64;
1868: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1869: Function MaxFloat( const X, Y : Float) : Float
1870: Function MaxFloatArray( const B : TDynFloatArray) : Float
1871: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1872: function MaxIntValue(const Data: array of Integer):Integer)
1873: Function MaxJ( const B1, B2 : Byte) : Byte;
1874: function MaxPath: string;
1875: function MaxValue(const Data: array of Double): Double)
1876: Function MaxCalc( const Formula : string) : Extended  //math expression parser
1877: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1878: function MD5(const fileName: string): string;
1879: Function Mean( const Data : array of Double) : Extended
1880: Function Median( const X : TDynFloatArray) : Float
1881: Function Memory : Pointer
1882: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1883: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1884: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1885: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1886: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1887: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1888: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1889: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1890: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1891: Function MibToId( Mib : string) : string
1892: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1893: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1894: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1895: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1896: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1897: Procedure GetMidiOutputs( const List : TStrings)
1898: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
1899: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
1900: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1901: Function MIDINoteToStr( Note : TMIDINote) : string
1902: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1903: Procedure GetMidiOutputs( const List : TStrings)
1904: Procedure MidiOutCheck( Code : MMResult)
1905: Procedure MidiInCheck( Code : MMResult)
1906: Function MilliSecondOf( const AValue : TDateTime) : Word
1907: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1908: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1909: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1910: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1911: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1912: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1913: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1914: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1915: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1916: Function milliToDateTime( MilliSecond : LongInt) : TDateTime');
1917: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1918: Function millis: int64;
1919: Function Min( AValueOne, AValueTwo : Integer) : Integer
1920: Function Min1( const B1, B2 : Shortint) : Shortint;
1921: Function Min2( const B1, B2 : Smallint) : Smallint;
1922: Function Min3( const B1, B2 : Word) : Word;
1923: Function Min4( const B1, B2 : Integer) : Integer;
1924: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1925: Function Min6( const B1, B2 : Int64) : Int64;
```

```
1926: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1927: Function MinClientRect : TRect;
1928: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1929: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1930: Function MinFloat( const X, Y : Float) : Float
1931: Function MinFloatArray( const B : TDynFloatArray) : Float
1932: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1933: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1934: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1935: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1936: Function MinIntValue( const Data : array of Integer) : Integer
1937: function MinIntValue(const Data: array of Integer):Integer)
1938: Function MinJ( const B1, B2 : Byte) : Byte;
1939: Function MinuteOf( const AValue : TDateTime) : Word
1940: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1941: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1942: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1943: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1944: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1945: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1946: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1947: Function MinValue( const Data : array of Double) : Double
1948: function MinValue(const Data: array of Double): Double)
1949: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1950: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1951: Function ModFloat( const X, Y : Float) : Float
1952: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1953: Function Modify( const Key : string; Value : Integer) : Boolean
1954: Function ModuleCacheID : Cardinal
1955: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1956: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1957: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1958: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1959: Function MonthOf( const AValue : TDateTime) : Word
1960: Function MonthOfTheYear( const AValue : TDateTime) : Word
1961: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1962: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1963: Function MonthStr( DateTime : TDateTime) : string
1964: Function MouseCoord( X, Y : Integer) : TGridCoord
1965: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1966: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1967: Function MoveNext : Boolean
1968: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1969: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1970: Function Name : string
1971: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1972: function NetworkVolume(DriveChar: Char): string
1973: Function NEWBOTTOMLINE : INTEGER
1974: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1975: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1976: Function NEWLINE : TMENUITEM
1977: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1978: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1979: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1980: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1981: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1982: Function NEWTOPLINE : INTEGER
1983: Function Next : TIdAuthWhatsNext
1984: Function NextCharIndex( S : String; Index : Integer) : Integer
1985: Function NextRecordSet : TCustomSQLDataSet
1986: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1987: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1988: Function NextToken : Char
1989: Function nextToken : WideString
1990: function NextToken:Char
1991: Function Norm( const Data : array of Double) : Extended
1992: Function NormalizeAngle( const Angle : Extended) : Extended
1993: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1994: Function NormalizeRect( const Rect : TRect) : TRect
1995: function NormalizeRect(const Rect: TRect): TRect;
1996: Function Now : TDateTime
1997: function Now2: tDateTime
1998: Function NumProcessThreads : integer
1999: Function NumThreadCount : integer
2000: Function NthDayOfWeek( const AValue : TDateTime) : Word
2001: Function NtProductType : TNtProductType
2002: Function NtProductTypeString : string
2003: function Null: Variant;
2004: Function NullPoint : TPoint
2005: Function NullRect : TRect
2006: Function Null2Blank(aString:String):String;
2007: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2008: Function NumIP : integer
```

```
2009: function Odd(x: Longint): boolean;
2010: Function OffsetFromUTC : TDateTime
2011: Function OffsetPoint( const P, Offset : TPoint) : TPoint
2012: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
2013: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
2014: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
2015: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
2016: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
2017: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
2018: function OpenBit:Integer
2019: Function OpenDatabase : TDatabase
2020: Function OpenDatabase( const DatabaseName : string) : TDatabase
2021: Procedure OpenDir(adir: string);
2022: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
2023: Function OpenMap(const Data: string): boolean;
2024: Function OpenMapX(const Data: string): boolean;
2025: Function OpenObject( Value : PChar) : Boolean;
2026: Function OpenObject1( Value : string) : Boolean;
2027: Function OpenSession( const SessionName : string) : TSession
2028: Function OpenVolume( const Drive : Char) : THandle
2029: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
2030: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
2031: Function OrdToBinary( const Value : Byte) : string;
2032: Function OrdToBinary1( const Value : Shortint) : string;
2033: Function OrdToBinary2( const Value : Smallint) : string;
2034: Function OrdToBinary3( const Value : Word) : string;
2035: Function OrdToBinary4( const Value : Integer) : string;
2036: Function OrdToBinary5( const Value : Cardinal) : string;
2037: Function OrdToBinary6( const Value : Int64) : string;
2038: Function OSCheck( RetVal : Boolean) : Boolean
2039: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
2040: Function OSIdentToString( const OSIdent : DWORD) : string
2041: Function Output: Text)
2042: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
2043: Function Owner : TCustomListView
2044: function Owner : TPersistent
2045: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2046: Function PadL( pStr : String; pLth : integer) : String
2047: Function Padl(s : AnyString;I : longInt) : AnyString;
2048: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
2049: Function PadR( pStr : String; pLth : integer) : String
2050: Function Padr(s : AnyString;I : longInt) : AnyString;
2051: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2052: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2053: Function Padz(s : AnyString;I : longInt) : AnyString;
2054: Function PaethPredictor( a, b, c : Byte) : Byte
2055: Function PARAMBYNAME( const VALUE : String) : TPARAM
2056: Function ParamByName( const Value : WideString) : TParameter
2057: Function ParamCount: Integer
2058: Function ParamsEncode( const ASrc : string) : string
2059: function ParamStr(Index: Integer): string)
2060: Function ParseDate( const DateStr : string) : TDateTime
2061: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2062: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2063: Function PathAddExtension( const Path, Extension : string) : string
2064: Function PathAddSeparator( const Path : string) : string
2065: Function PathAppend( const Path, Append : string) : string
2066: Function PathBuildRoot( const Drive : Byte) : string
2067: Function PathCanonicalize( const Path : string) : string
2068: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2069: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2070: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2071: Function PathEncode( const ASrc : string) : string
2072: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2073: Function PathExtractFileNameNoExt( const Path : string) : string
2074: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2075: Function PathGetDepth( const Path : string) : Integer
2076: Function PathGetLongName( const Path : string) : string
2077: Function PathGetLongName2( Path : string) : string
2078: Function PathGetShortName( const Path : string) : string
2079: Function PathIsAbsolute( const Path : string) : Boolean
2080: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2081: Function PathIsDiskDevice( const Path : string) : Boolean
2082: Function PathIsUNC( const Path : string) : Boolean
2083: Function PathRemoveExtension( const Path : string) : string
2084: Function PathRemoveSeparator( const Path : string) : string
2085: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2086: Function Peek : Pointer
2087: Function Peek : TObject
2088: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2089: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2090: function Permutation(npr, k: integer): extended;
2091: function PermutationInt(npr, k: integer): Int64;
2092: Function PermutationJ( N, R : Cardinal) : Float
2093: Function Pi : Extended;
2094: Function PiE : Extended;
2095: Function PixelsToDialogsX( const Pixels : Word) : Word
```

```
2096: Function PixelsToDialogsY( const Pixels : Word) : Word
2097: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2098: Function Point( X, Y : Integer) : TPoint
2099: function Point(X, Y: Integer): TPoint)
2100: Function PointAssign( const X, Y : Integer) : TPoint
2101: Function PointDist( const P1, P2 : TPoint) : Double;
2102: function PointDist(const P1,P2: TFloatPoint): Double;
2103: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2104: function PointDist2(const P1,P2: TPoint): Double;
2105: Function PointEqual( const P1, P2 : TPoint) : Boolean
2106: Function PointIsNull( const P : TPoint) : Boolean
2107: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2108: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2109: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2110: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2111: Function Pop : Pointer
2112: Function Pop : TObject
2113: Function PopnStdDev( const Data : array of Double) : Extended
2114: Function PopnVariance( const Data : array of Double) : Extended
2115: Function PopulationVariance( const X : TDynFloatArray) : Float
2116: function Pos(SubStr, S: AnyString): Longint;
2117: Function PosEqual( const Rect : TRect) : Boolean
2118: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2119: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2120: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2121: Function Post1( AURL : string; const ASource : TStrings) : string;
2122: Function Post2( AURL : string; const ASource : TStream) : string;
2123: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2124: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2125: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2126: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2127: Function Power( const Base, Exponent : Extended) : Extended
2128: Function PowerBig(aval, n:integer): string;
2129: Function PowerIntJ( const X : Float; N : Integer) : Float;
2130: Function PowerJ( const Base, Exponent : Float) : Float;
2131: Function PowerOffOS : Boolean
2132: Function PreformatDateString( Ps : string) : string
2133: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2134: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2135: Function Printer : TPrinter
2136: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2137: Function ProcessResponse : TIdHTTPWhatsNext
2138: Function ProduceContent : string
2139: Function ProduceContentFromStream( Stream : TStream) : string
2140: Function ProduceContentFromString( const S : string) : string
2141: Function ProgIDToClassID(const ProgID: string): TGUID;
2142: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2143: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2144: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2145: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2146: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2147: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2148: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2149: Function Push( AItem : Pointer) : Pointer
2150: Function Push( AObject : TObject) : TObject
2151: Function Put1( AURL : string; const ASource : TStream) : string;
2152: Function Pythagoras( const X, Y : Extended) : Extended
2153: Function queryDLLInterface( var queryList : TStringList) : TStringList
2154: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2155: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2156: Function queryPerformanceCounter2(mse: int64): int64;
2157: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2158: //Function QueryPerformanceFrequency(mse: int64): boolean;
2159: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2160: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2161: Procedure QueryPerformanceCounter1(var aC: Int64);
2162: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2163: Function Quote( const ACommand : String) : SmallInt
2164: Function QuotedStr( S : string) : string
2165: Function RadToCycle( const Radians : Extended) : Extended
2166: Function RadToDeg( const Radians : Extended) : Extended
2167: Function RadToDeg( const Value : Extended) : Extended;
2168: Function RadToDeg1( const Value : Double) : Double;
2169: Function RadToDeg2( const Value : Single) : Single;
2170: Function RadToGrad( const Radians : Extended) : Extended
2171: Function RadToGrad( const Value : Extended) : Extended;
2172: Function RadToGrad1( const Value : Double) : Double;
2173: Function RadToGrad2( const Value : Single) : Single;
2174: Function RandG( Mean, StdDev : Extended) : Extended
2175: function Random(const ARange: Integer): Integer;
2176: function random2(a: integer): double
2177: function RandomE: Extended;
2178: function RandomF: Extended;
2179: Function RandomFrom( const AValues : array of string) : string;
2180: Function RandomRange( const AFrom, ATo : Integer) : Integer
2181: function randSeed: longint
```

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

```
2269: Function RemoveDir( Dir : string) : Boolean
2270: function RemoveDir(const Dir: string): Boolean)
2271: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2272: Function RemoveFileExt( const FileName : string) : string
2273: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2274: Function RenameFile( OldName, NewName : string) : Boolean
2275: function RenameFile(const OldName: string; const NewName: string): Boolean)
2276: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2277: Function ReplaceText( const AText, AFromText, AToText : string) : string
2278: Function Replicate(c : char;I : longInt) : String;
2279: Function Request : TWebRequest
2280: Function ResemblesText( const AText, AOther : string) : Boolean
2281: Function Reset : Boolean
2282: function Reset2(mypath: string):string;
2283: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2284: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2285: Function Response : TWebResponse
2286: Function ResumeSupported : Boolean
2287: Function RETHINKHOTKEYS : BOOLEAN
2288: Function RETHINKLINES : BOOLEAN
2289: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2290: Function RetrieveCurrentDir : string
2291: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2292: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2293: Function RetrieveMailBoxSize : integer
2294: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2295: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2296: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2297: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2298: Function ReverseBits( Value : Byte) : Byte;
2299: Function ReverseBits1( Value : Shortint) : Shortint;
2300: Function ReverseBits2( Value : Smallint) : Smallint;
2301: Function ReverseBits3( Value : Word) : Word;
2302: Function ReverseBits4( Value : Cardinal) : Cardinal;
2303: Function ReverseBits4( Value : Integer) : Integer;
2304: Function ReverseBits5( Value : Int64) : Int64;
2305: Function ReverseBytes( Value : Word) : Word;
2306: Function ReverseBytes1( Value : Smallint) : Smallint;
2307: Function ReverseBytes2( Value : Integer) : Integer;
2308: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2309: Function ReverseBytes4( Value : Int64) : Int64;
2310: Function ReverseString( const AText : string) : string
2311: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
     HostName:String):Bool;
2312: Function Revert : HResult
2313: Function RGB(R,G,B: Byte): TColor;
2314: Function RGB2BGR( const Color : TColor) : TColor
2315: Function RGB2TColor( R, G, B : Byte) : TColor
2316: Function RGBToWebColorName( RGB : Integer) : string
2317: Function RGBToWebColorStr( RGB : Integer) : string
2318: Function RgbToHtml( Value : TColor) : string
2319: Function HtmlToRgb(const Value: string): TColor;
2320: Function RightStr( const AStr : String; Len : Integer) : String
2321: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2322: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2323: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2324: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2325: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2326: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2327: Function Round(e : Extended) : Longint;
2328: Function Round64(e: extended): Int64;
2329: Function RoundAt( const Value : string; Position : SmallInt) : string
2330: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2331: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;');
2332: Function SimpleRoundTo(const AValue: Extended; const ADigit: TRoundToRange): Extended;');
2333: Function RoundFrequency( const Frequency : Integer) : Integer
2334: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2335: Function RoundPoint( const X, Y : Double) : TPoint
2336: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2337: Function RowCount : Integer
2338: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2339: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2340: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2341: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2342: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2343: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2344: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2345: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2346: Function RunByteCode(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2347: Function RunCompiledScript2(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2348: Function S_AddBackSlash( const ADirName : string) : string
2349: Function S_AllTrim( const cStr : string) : string
2350: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2351: Function S_Cut( const cStr : string; const iLen : integer) : string
2352: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2353: Function S_DirExists( const ADir : string) : Boolean
2354: Function S_Empty( const cStr : string) : boolean
2355: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2356: Function S_LargeFontsActive : Boolean
```

```
2357: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2358: Function S_LTrim( const cStr : string) : string
2359: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2360: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2361: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2362: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2363: Function S_RTrim( const cStr : string) : string
2364: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2365: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2366: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2367: Function S_Space( const iLen : integer) : String
2368: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2369: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2370: Function S_StrCRC32( const Text : string) : LongWORD
2371: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2372: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2373: Function S_StringtoUTF_8( const AString : string) : string
2374: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2375: function S_StrToReal(const cStr: string; var R: Double): Boolean
2376: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2377: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2378: Function S_UTF_8ToString( const AString : string) : string
2379: Function S_WBox( const AText : string) : integer
2380: Function SameDate( const A, B : TDateTime) : Boolean
2381: function SameDate(const A, B: TDateTime): Boolean;
2382: Function SameDateTime( const A, B : TDateTime) : Boolean
2383: function SameDateTime(const A, B: TDateTime): Boolean;
2384: Function SameFileName( S1, S2 : string) : Boolean
2385: Function SameText( S1, S2 : string) : Boolean
2386: function SameText(const S1: string; const S2: string): Boolean)
2387: Function SameTime( const A, B : TDateTime) : Boolean
2388: function SameTime(const A, B: TDateTime): Boolean;
2389: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2390: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2391: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2392: Function SampleVariance( const X : TDynFloatArray) : Float
2393: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2394: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2395: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2396: Function SaveToFile( const AFileName : TFileName) : Boolean
2397: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2398: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2399: Function ScanF(const aformat: String; const args: array of const): string;
2400: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2401: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2402: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2403: function SearchRecattr: integer;
2404: function SearchRecExcludeAttr: integer;
2405: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2406: function SearchRecname: string;
2407: function SearchRecsize: integer;
2408: function SearchRecTime: integer;
2409: Function Sec( const X : Extended) : Extended
2410: Function Secant( const X : Extended) : Extended
2411: Function SecH( const X : Extended) : Extended
2412: Function SecondOf( const AValue : TDateTime) : Word
2413: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2414: Function SecondOfTheHour( const AValue : TDateTime) : Word
2415: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2416: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2417: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2418: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2419: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2420: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2421: Function SectionExists( const Section : string) : Boolean
2422: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2423: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2424: function Seek(Offset:LongInt;Origin:Word):LongInt
2425: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2426: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2427: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2428: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2429: Function SendBuf( var Buf, Count : Integer) : Integer
2430: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2431: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2432: Function SendKey( AppName : string; Key : Char) : Boolean
2433: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2434: Function SendStream( AStream : TStream) : Boolean
2435: Function SendStreamThenDrop( AStream : TStream) : Boolean
2436: Function SendText( const S : string) : Integer
2437: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2438: Function SendSerialText(Data: String): cardinal
2439: Function Sent : Boolean
2440: Function ServicesFilePath: string
2441: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2442: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
```

```
2443: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2444: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2445: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2446: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2447: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2448: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2449: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2450: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2451: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2452: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2453: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2454: Function SetCurrentDir( Dir : string) : Boolean
2455: function SetCurrentDir(const Dir: string): Boolean)
2456: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2457: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2458: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2459: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2460: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2461: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2462: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2463: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2464: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2465: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2466: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2467: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2468: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2469: Function SetLocalTime( Value : TDateTime) : boolean
2470: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2471: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2472: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2473: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2474: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2475: Function SetSize( libNewSize : Longint) : HResult
2476: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2477: Function Sgn( const X : Extended) : Integer
2478: function SHA1(const fileName: string): string;
2479: function SHA256(astr: string; amode: char): string)
2480: function SHA512(astr: string; amode: char): string)
2481: Function ShareMemoryManager : Boolean
2482: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2483: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2484: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2485: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2486: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2487: function ShortDateFormat: string;
2488: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2489: function ShortTimeFormat: string;
2490: function SHOWMODAL:INTEGER
2491: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
      TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult');
2492: Function
      ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2493: function ShowWindow(C1: HWND; C2: integer): boolean;
2494: procedure ShowMemory     //in Dialog
2495: function ShowMemory2: string;
2496: Function ShutDownOS : Boolean
2497: Function Signe( const X, Y : Extended) : Extended
2498: Function Sign( const X : Extended) : Integer
2499: Function Sin(e : Extended) : Extended;
2500: Function sinc( const x : Double) : Double
2501: Function SinJ( X : Float) : Float
2502: Function Size( const AFileName : String) : Integer
2503: function SizeOf: Longint;
2504: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2505: function SlashSep(const Path, S: String): String
2506: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2507: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2508: Function SmallPoint(X, Y: Integer): TSmallPoint)
2509: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2510: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2511: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2512: Function SoundexProc( const AText, AOther : string) : Boolean
2513: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2514: Function SoundexWord( const AText : string) : Word
2515: Function SourcePos : Longint
2516: function SourcePos:LongInt
2517: Function Split0( Str : string; const substr : string) : TStringList
2518: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2519: Function SQLRequiresParams( const SQL : WideString) : Boolean
2520: Function Sqr(e : Extended) : Extended;
2521: Function Sqrt(e : Extended) : Extended;
2522: Function StartIP : String
2523: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2524: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2525: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2526: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2527: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2528: Function StartOfAYear( const AYear : Word) : TDateTime
```

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

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

```
2707: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2708: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2709: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2710: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2711: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2712: Function TenToY( const Y : Float) : Float
2713: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2714: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2715: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2716: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2717: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2718: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2719: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2720: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2721: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2722: Function TestBits( const Value, Mask : Byte) : Boolean;
2723: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2724: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2725: Function TestBits3( const Value, Mask : Word) : Boolean;
2726: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2727: Function TestBits5( const Value, Mask : Integer) : Boolean;
2728: Function TestBits6( const Value, Mask : Int64) : Boolean;
2729: Function TestFDIVInstruction : Boolean
2730: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2731: Function TextExtent( const Text : string) : TSize
2732: function TextHeight(Text: string): Integer;
2733: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2734: Function TextStartsWith( const S, SubS : string) : Boolean
2735: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2736: Function ConvInteger(i : integer):string;
2737: Function IntegerToText(i : integer):string;
2738: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2739: function TextWidth(Text: string): Integer;
2740: Function ThreadCount : integer
2741: function ThousandSeparator: char;
2742: Function Ticks : Cardinal
2743: Function Time : TDateTime
2744: function Time: TDateTime;
2745: function TimeGetTime: int64;
2746: Function TimeOf( const AValue : TDateTime) : TDateTime
2747: function TimeSeparator: char;
2748: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2749: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2750: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2751: Function TimeToStr( DateTime : TDateTime) : string;
2752: function TimeToStr(const DateTime: TDateTime): string;
2753: Function TimeZoneBias : TDateTime
2754: Function ToCommon( const AValue : Double) : Double
2755: function ToCommon(const AValue: Double): Double;
2756: Function Today : TDateTime
2757: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2758: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2759: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2760: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2761: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2762: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2763: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2764: function TokenComponentIdent:String
2765: Function TokenFloat : Extended
2766: function TokenFloat:Extended
2767: Function TokenInt : Longint
2768: function TokenInt:LongInt
2769: Function TokenString : string
2770: function TokenString:String
2771: Function TokenSymbolIs( const S : string) : Boolean
2772: function TokenSymbolIs(S:String):Boolean
2773: Function Tomorrow : TDateTime
2774: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2775: Function ToString : string
2776: Function TotalVariance( const Data : array of Double) : Extended
2777: Function Trace2( AURL : string) : string;
2778: Function TrackMenu( Button : TToolButton) : Boolean
2779: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2780: Function TranslateURI( const URI : string) : string
2781: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2782: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2783: Function Trim( S : string) : string;
2784: Function Trim( S : WideString) : WideString;
2785: Function Trim(s : AnyString) : AnyString;
2786: Function TrimAllOf( ATrim, AText : String) : String
2787: Function TrimLeft( S : string) : string;
2788: Function TrimLeft( S : WideString) : WideString;
2789: function TrimLeft(const S: string): string)
```

```
2790: Function TrimRight( S : string) : string;
2791: Function TrimRight( S : WideString) : WideString;
2792: function TrimRight(const S: string): string)
2793: function TrueBoolStrs: array of string
2794: Function Trunc(e : Extended) : Longint;
2795: Function Trunc64(e: extended): Int64;
2796: Function TruncPower( const Base, Exponent : Float) : Float
2797: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2798: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2799: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2800: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2801: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2802: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
2803: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2804: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2805: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2806: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2807: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2808: Function TryLock : Boolean
2809: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2810: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2811: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2812: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2813: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2814: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2815: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2816: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2817: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2818: function TryStrToBool(const S: string; out Value: Boolean): Boolean;
2819: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2820: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2821: Function TwoToY( const Y : Float) : Float
2822: Function UCS4StringToWideString( const S : UCS4String) : WideString
2823: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2824: function Unassigned: Variant;
2825: Function UndoLastChange( FollowChange : Boolean) : Boolean
2826: function UniCodeToStr(Value: string): string;
2827: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2828: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2829: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2830: Function UnixPathToDosPath( const Path : string) : string
2831: Function UnixToDateTime( const AValue : Int64) : TDateTime
2832: function UnixToDateTime(U: Int64): TDateTime;
2833: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2834: Function UnlockResource( ResData : HGLOBAL) : LongBool
2835: Function UnlockVolume( var Handle : THandle) : Boolean
2836: Function UnMaskString( Mask, Value : String) : String
2837: function UpCase(ch : Char ) : Char;
2838: Function UpCaseFirst( const AStr : string) : string
2839: Function UpCaseFirstWord( const AStr : string) : string
2840: Function UpdateAction( Action : TBasicAction) : Boolean
2841: Function UpdateKind : TUpdateKind
2842: Function UPDATESTATUS : TUPDATESTATUS
2843: Function UpperCase( S : string) : string
2844: Function Uppercase(s : AnyString) : AnyString;
2845: Function URLDecode( ASrc : string) : string
2846: Function URLEncode( const ASrc : string) : string
2847: Function UseRightToLeftAlignment : Boolean
2848: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2849: Function UseRightToLeftReading : Boolean
2850: Function UTF8CharLength( Lead : Char) : Integer
2851: Function UTF8CharSize( Lead : Char) : Integer
2852: Function UTF8Decode( const S : UTF8String) : WideString
2853: Function UTF8Encode( const WS : WideString) : UTF8String
2854: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2855: Function Utf8ToAnsi( const S : UTF8String) : string
2856: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2857: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2858: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2859: Function ValidParentForm(control: TControl): TForm
2860: Function Value : Variant
2861: Function ValueExists( const Section, Ident : string) : Boolean
2862: Function ValueOf( const Key : string) : Integer
2863: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2864: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2865: Function VarArrayFromStrings( Strings : TStrings) : Variant
2866: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2867: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2868: Function VarFMTBcd : TVarType
2869: Function VarFMTBcdCreate1 : Variant;
2870: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2871: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2872: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2873: Function Variance( const Data : array of Double) : Extended
2874: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2875: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
```

```
2876: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2877: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2878: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2879: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2880: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2881: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2882: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2883: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2884: Function VariantNeg( const V1 : Variant) : Variant
2885: Function VariantNot( const V1 : Variant) : Variant
2886: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2887: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2888: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2889: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2890: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2891: function VarIsEmpty(const V: Variant): Boolean;
2892: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2893: function VarIsNull(const V: Variant): Boolean;
2894: Function VarToBcd( const AValue : Variant) : TBcd
2895: function VarType(const V: Variant): TVarType;
2896: Function VarType( const V : Variant) : TVarType
2897: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2898: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2899: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2900: Function VarIsByRef( const V : Variant) : Boolean
2901: Function VarIsEmpty( const V : Variant) : Boolean
2902: Procedure VarCheckEmpty( const V : Variant)
2903: Function VarIsNull( const V : Variant) : Boolean
2904: Function VarIsClear( const V : Variant) : Boolean
2905: Function VarIsCustom( const V : Variant) : Boolean
2906: Function VarIsOrdinal( const V : Variant) : Boolean
2907: Function VarIsFloat( const V : Variant) : Boolean
2908: Function VarIsNumeric( const V : Variant) : Boolean
2909: Function VarIsStr( const V : Variant) : Boolean
2910: Function VarToStr( const V : Variant) : string
2911: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2912: Function VarToWideStr( const V : Variant) : WideString
2913: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2914: Function VarToDateTime( const V : Variant) : TDateTime
2915: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2916: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2917: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2918: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2919: Function VarSameValue( const A, B : Variant) : Boolean
2920: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2921: Function VarIsEmptyParam( const V : Variant) : Boolean
2922: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2923: Function VarIsError1( const V : Variant) : Boolean;
2924: Function VarAsError( AResult : HRESULT) : Variant
2925: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2926: Function VarIsArray( const A : Variant) : Boolean;
2927: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2928: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2929: Function VarArrayOf( const Values : array of Variant) : Variant
2930: Function VarArrayRef( const A : Variant) : Variant
2931: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2932: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2933: Function VarArrayDimCount( const A : Variant) : Integer
2934: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2935: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2936: Function VarArrayLock( const A : Variant) : ___Pointer
2937: Procedure VarArrayUnlock( const A : Variant)
2938: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2939: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2940: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2941: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2942: Function Unassigned : Variant
2943: Function Null : Variant
2944: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2945: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2946: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2947: function VectorDot(const V1,V2: TFloatPoint): Double;
2948: Function VectorLengthSqr( const V : TFloatPoint) : Double
2949: function VectorLengthSqr(const V: TFloatPoint): Double;
2950: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2951: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2952: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2953: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2954: Function Verify( AUserName : String) : String
2955: Function Versine( X : Float) : Float
2956: function VersionCheck: boolean;
2957: function VersionCheckAct: string;
2958: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2959: Function VersionLanguageName( const LangId : Word) : string
2960: Function VersionResourceAvailable( const FileName : string) : Boolean
2961: Function Visible : Boolean
2962: function VolumeID(DriveChar: Char): string
2963: Function WaitFor( const AString : string) : string
2964: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
```

```
2965: Function WaitFor1 : TWaitResult;
2966: Function WaitForData( Timeout : Longint) : Boolean
2967: Function WebColorNameToColor( WebColorName : string) : TColor
2968: Function WebColorStrToColor( WebColor : string) : TColor
2969: Function WebColorToRGB( WebColor : Integer) : Integer
2970: Function wGet(aURL, afile: string): boolean;'
2971: Function wGet2(aURL, afile: string): boolean;'  //without file open
2972: Function wGetX(aURL, afile: string): boolean;
2973: Function wGetX2(aURL, afile: string): boolean;'  //without file open
2974: Function WebGet(aURL, afile: string): boolean;'
2975: Function WebExists: boolean;   //alias to isinternet
2976: Function WeekOf( const AValue : TDateTime) : Word
2977: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2978: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2979: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2980: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2981: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2982: Function WeeksInAYear( const AYear : Word) : Word
2983: Function WeeksInYear( const AValue : TDateTime) : Word
2984: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2985: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2986: Function WideCat( const x, y : WideString) : WideString
2987: Function WideCompareStr( S1, S2 : WideString) : Integer
2988: Function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2989: Function WideCompareText( S1, S2 : WideString) : Integer
2990: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2991: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2992: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2993: Function WideEqual( const x, y : WideString) : Boolean
2994: function WideFormat(const Format: WideString; const Args: array of const): WideString)
2995: Function WideGreater( const x, y : WideString) : Boolean
2996: Function WideLength( const src : WideString) : Integer
2997: Function WideLess( const x, y : WideString) : Boolean
2998: Function WideLowerCase( S : WideString) : WideString
2999: function WideLowerCase(const S: WideString): WideString)
3000: Function WidePos( const src, sub : WideString) : Integer
3001: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
3002: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
3003: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
3004: Function WideSameStr( S1, S2 : WideString) : Boolean
3005: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
3006: Function WideSameText( S1, S2 : WideString) : Boolean
3007: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
3008: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
3009: Function WideStringToUCS4String( const S : WideString) : UCS4String
3010: Function WideUpperCase( S : WideString) : WideString
3011: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
3012: function Win32Check(RetVal: boolean): boolean)
3013: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
3014: Function Win32RestoreFile( const FileName : string) : Boolean
3015: Function Win32Type : TIdWin32Type
3016: Function WinColor( const Color32 : TColor32) : TColor
3017: function winexec(FileName: pchar; showCmd: integer): integer;
3018: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3019: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3020: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
3021: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
3022: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
3023: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
3024: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
3025: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
3026: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
3027: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
3028: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
3029: Function WordToStr( const Value : Word) : String
3030: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
3031: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
3032: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
3033: Function WorkArea : Integer
3034: Function WrapText( Line : string; MaxCol : Integer) : string;
3035: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
3036: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3037: function Write(Buffer:String;Count:LongInt):LongInt
3038: Function WriteClient( var Buffer, Count : Integer) : Integer
3039: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
3040: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
3041: Function WriteString( const AString : string) : Boolean
3042: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3043: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3044: Function wsprintf( Output : PChar; Format : PChar) : Integer
3045: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3046: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3047: Function XorDecode( const Key, Source : string) : string
3048: Function XorEncode( const Key, Source : string) : string
3049: Function XorString( const Key, Src : ShortString) : ShortString
3050: Function Yield : Bool
3051: Function YearOf( const AValue : TDateTime) : Word
3052: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3053: Function YearSpan( const ANow, AThen : TDateTime) : Double
```

```
3054: Function Yesterday : TDateTime
3055: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3056: Function( const Name : string; Proc : TUserFunction)
3057: Function using Special_Scholz from 3.8.5.0
3058: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3059: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3060: Function FloatToTime2Dec(value:Extended):Extended;
3061: Function MinToStd(value:Extended):Extended;
3062: Function MinToStdAsString(value:Extended):String;
3063: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3064: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3065: Function Round2Dec (zahl:Extended):Extended;
3066: Function GetAngle(x,y:Extended):Double;
3067: Function AddAngle(a1,a2:Double):Double;
3068:
3069: ************************************************************************
3070: unit uPSI_StText;
3071: ************************************************************************
3072: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3073: Function TextFileSize( var F : TextFile) : LongInt
3074: Function TextPos( var F : TextFile) : LongInt
3075: Function TextFlush( var F : TextFile) : Boolean
3076:
3077: ************************************************************************
3078: from  JvVCLUtils;
3079: ************************************************************************
3080: { Windows resources (bitmaps and icons) VCL-oriented routines }
3081: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3082: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
      DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3083: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
      Bitmap: TBitmap; TransparentColor: TColor);
3084: function MakeBitmap(ResID: PChar): TBitmap;
3085: function MakeBitmapID(ResID: Word): TBitmap;
3086: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3087: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3088: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3089: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3090:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3091: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3092: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3093: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3094: {$IFDEF WIN32}
3095: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3096:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3097: {$ENDIF}
3098: function MakeIcon(ResID: PChar): TIcon;
3099: function MakeIconID(ResID: Word): TIcon;
3100: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3101: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3102: {$IFDEF WIN32}
3103: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3104: {$ENDIF}
3105: { Service routines }
3106: procedure NotImplemented;
3107: procedure ResourceNotFound(ResID: PChar);
3108: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3109: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3110: function PaletteColor(Color: TColor): Longint;
3111: function WidthOf(R: TRect): Integer;
3112: function HeightOf(R: TRect): Integer;
3113: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3114: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3115: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3116: procedure Delay(MSecs: Longint);
3117: procedure CenterControl(Control: TControl);
3118: Function PaletteEntries( Palette : HPALETTE) : Integer
3119: Function WindowClassName( Wnd : HWND) : string
3120: Function ScreenWorkArea : TRect
3121: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3122: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3123: Procedure ActivateWindow( Wnd : HWND)
3124: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3125: Procedure CenterWindow( Wnd : HWND)
3126: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3127: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3128: Function DialogsToPixelsX( Dlgs : Word) : Word
3129: Function DialogsToPixelsY( Dlgs : Word) : Word
3130: Function PixelsToDialogsX( Pixs : Word) : Word
3131: Function PixelsToDialogsY( Pixs : Word) : Word
3132: {$IFDEF WIN32}
3133: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3134: function MakeVariant(const Values: array of Variant): Variant;
3135: {$ENDIF}
3136: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3137: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3138: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3139: {$IFDEF CBUILDER}
3140: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
```

```
3141: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3142: {$ELSE}
3143: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3144: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3145: {$ENDIF CBUILDER}
3146: function IsForegroundTask: Boolean;
3147: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3148: function GetAveCharSize(Canvas: TCanvas): TPoint;
3149: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3150: procedure FreeUnusedOle;
3151: procedure Beep;
3152: function GetWindowsVersionJ: string;
3153: function LoadDLL(const LibName: string): THandle;
3154: function RegisterServer(const ModuleName: string): Boolean;
3155: {$IFNDEF WIN32}
3156: function IsLibrary: Boolean;
3157: {$ENDIF}
3158: { Gradient filling routine }
3159: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3160: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
     TFillDirection; Colors: Byte);
3161: { String routines }
3162: function GetEnvVar(const VarName: string): string;
3163: function AnsiUpperFirstChar(const S: string): string;
3164: function StringToPChar(var S: string): PChar;
3165: function StrPAlloc(const S: string): PChar;
3166: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3167: function DropT(const S: string): string;
3168: { Memory routines }
3169: function AllocMemo(Size: Longint): Pointer;
3170: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3171: procedure FreeMemo(var fpBlock: Pointer);
3172: function GetMemoSize(fpBlock: Pointer): Longint;
3173: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3174: {$IFNDEF COMPILER5_UP}
3175: procedure FreeAndNil(var Obj);
3176: {$ENDIF}
3177: // from PNGLoader
3178: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3179: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3180: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3181: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3182: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
     TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3183:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3184:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3185:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3186:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3187:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3188:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3189:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3190:  Procedure SetImeName( Name : TImeName)
3191:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3192:  Function Imm32GetContext( hWnd : HWND) : HIMC
3193:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3194:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3195:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3196:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3197: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3198:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3199:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3200:  Function Imm32IsIME( hKl : longword) : Boolean
3201:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3202:  Procedure DragDone( Drop : Boolean)
3203:
3204:
3205: //****************************************added  from jvjvclutils
3206: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3207: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3208: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3209: function IsPositiveResult(Value: TModalResult): Boolean;
3210: function IsNegativeResult(Value: TModalResult): Boolean;
3211: function IsAbortResult(const Value: TModalResult): Boolean;
3212: function StripAllFromResult(const Value: TModalResult): TModalResult;
3213: // returns either BrightColor or DarkColor depending on the luminance of AColor
3214: // This function gives the same result (AFAIK) as the function used in Windows to
3215: // calculate the desktop icon text color based on the desktop background color
3216: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3217: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3218:
3219: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3220:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3221:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3222:   var LinkName: string; Scale: Integer = 100); overload;
3223: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3224:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3225:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3226:   var LinkName: string; Scale: Integer = 100); overload;
3227: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
```

```
3228:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3229: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3230:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3231:   Scale: Integer = 100): string;
3232: function HTMLPlainText(const Text: string): string;
3233: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3234:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3235: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3236:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3237: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3238: function HTMLPrepareText(const Text: string): string;
3239:
3240: ****************************************** uPSI_JvAppUtils;
3241: Function GetDefaultSection( Component : TComponent) : string
3242: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3243: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3244: Function GetDefaultIniName : string
3245:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3246: Function GetDefaultIniRegKey : string
3247: Function FindForm( FormClass : TFormClass) : TForm
3248: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3249: Function ShowDialog( FormClass : TFormClass) : Boolean
3250:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3251: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3252: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3253: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3254: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3255: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3256: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3257: Function StrToIniStr( const Str : string) : string
3258: Function IniStrToStr( const Str : string) : string
3259: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3260: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3261: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3262: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3263: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3264: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3265: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3266: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3267: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3268: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3269: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3270: Procedure AppTaskbarIcons( AppOnly : Boolean)
3271: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3272: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3273: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3274: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3275: ****************************************** uPSI_JvDBUtils;
3276: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3277: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3278: Procedure RefreshQuery( Query : TDataSet)
3279: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3280: Function DataSetSectionName( DataSet : TDataSet) : string
3281: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3282: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3283: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
      Options: TLocateOptions) : Boolean
3284: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3285: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3286: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3287: Function ConfirmDelete : Boolean
3288: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3289: Procedure CheckRequiredField( Field : TField)
3290: Procedure CheckRequiredFields( const Fields : array of TField)
3291: Function DateToSQL( Value : TDateTime) : string
3292: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3293: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3294: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
      HighEmpty:Double;Inclusive:Bool):string
3295: Function StrMaskSQL( const Value : string) : string
3296: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3297: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3298: Procedure _DBError( const Msg : string)
3299:  Const('TrueExpr','String '0=0
3300:  Const('sdfStandard16','String '''"''mm''/''dd''/''yyyy''"''
3301:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3302:  Const('sdfOracle','String '"TO_DATE(''"dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3303:  Const('sdfInterbase','String '"CAST(''"mm"/"dd"/"yyyy"'' AS DATE)"
3304:  Const('sdfMSSQL','String '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3305:  AddTypeS('Largeint', 'Longint
3306:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3307:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3308:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3309:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3310:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3311: (*----------------------------------------------------------------------------*)
3312: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3313: begin
3314:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
```

```
3315:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3316:  Function JIniReadString( const FileName, Section, Line : string) : string
3317:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3318:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3319:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3320:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3321:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3322: end;
3323:
3324: (* === compile-time registration functions === *)
3325: (*----------------------------------------------------------------------------*)
3326: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3327: begin
3328:  'UnixTimeStart','LongInt'( 25569);
3329:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3330:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3331:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3332:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3333:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3334:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3335:  Function DayOfDate( const DateTime : TDateTime) : Integer
3336:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3337:  Function YearOfDate( const DateTime : TDateTime) : Integer
3338:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3339:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3340:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3341:  Function HourOfTime( const DateTime : TDateTime) : Integer
3342:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3343:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3344:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3345:  Function IsISOLongYear( const Year : Word) : Boolean;
3346:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3347:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3348:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3349:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3350:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3351:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3352:  Function JIsLeapYear( const Year : Integer) : Boolean;
3353:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3354:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3355:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3356:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3357:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3358:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3359:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3360:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3361:  Function HoursToMSecs( Hours : Integer) : Integer
3362:  Function MinutesToMSecs( Minutes : Integer) : Integer
3363:  Function SecondsToMSecs( Seconds : Integer) : Integer
3364:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3365:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3366:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3367:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3368:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3369:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3370:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3371:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3372:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3373:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3374:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3375:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3376:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3377:  Function DosDateTimeToStr( DateTime : Integer) : string
3378:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3379:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3380:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3381:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3382:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3383:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3384:  Function FileTimeToStr( const FileTime : TFileTime) : string
3385:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3386:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3387:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3388:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3389:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3390:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3391:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3392:   TJclUnixTime32', 'Longword
3393:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3394:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3395:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3396:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3397:  Function JNullStamp : TTimeStamp
3398:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3399:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3400:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3401:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3402:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3403:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
```

```
3404:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3405:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3406:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3407:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3408:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3409:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3410:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3411:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3412:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3413:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3414:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3415:   FindClass('TOBJECT'),'EJclDateTimeError
3416: end;
3417:
3418: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3419: begin
3420:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3421:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3422:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3423:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3424:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3425:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3426:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3427:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3428:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3429:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3430:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3431:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3432:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3433:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3434:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3435:  Function AbortShutDown : Boolean;
3436:  Function AbortShutDown1( const MachineName : string) : Boolean;
3437:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3438:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3439:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3440:   FindClass('TOBJECT'),'EJclCreateProcessError
3441:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3442:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
     Environment:PChar);
3443:   // with Add(EJclCreateProcessError) do
3444:  end;
3445:
3446:
3447: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3448: begin
3449:   //'AnsiSigns','Set').SetSet(['-', '+']);
3450:  'C1_UPPER','LongWord( $0001);
3451:  'C1_LOWER','LongWord( $0002);
3452:  'C1_DIGIT','LongWord').SetUInt( $0004);
3453:  'C1_SPACE','LongWord').SetUInt( $0008);
3454:  'C1_PUNCT','LongWord').SetUInt( $0010);
3455:  'C1_CNTRL','LongWord').SetUInt( $0020);
3456:  'C1_BLANK','LongWord').SetUInt( $0040);
3457:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3458:  'C1_ALPHA','LongWord').SetUInt( $0100);
3459:   AnsiChar', 'Char
3460:  Function StrIsAlpha( const S : AnsiString) : Boolean
3461:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3462:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3463:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3464:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3465:  Function StrIsDigit( const S : AnsiString) : Boolean
3466:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3467:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3468:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3469:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3470:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3471:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3472:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3473:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3474:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3475:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3476:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3477:  Function JStrLower( const S : AnsiString) : AnsiString
3478:  Procedure StrLowerInPlace( var S : AnsiString)
3479:  ///Procedure StrLowerBuff( S : PAnsiChar)
3480:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3481:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3482:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3483:  Function StrProper( const S : AnsiString) : AnsiString
3484:  //Procedure StrProperBuff( S : PAnsiChar)
3485:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3486:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3487:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3488:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3489:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3490:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3491:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
```

```
3492:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3493:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3494:  Function StrReverse( const S : AnsiString) : AnsiString
3495:  Procedure StrReverseInPlace( var S : AnsiString)
3496:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3497:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3498:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3499:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3500:  Function StrToHex( const Source : AnsiString) : AnsiString
3501:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3502:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3503:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3504:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3505:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
3506:  Function JStrUpper( const S : AnsiString) : AnsiString
3507:  Procedure StrUpperInPlace( var S : AnsiString)
3508:  //Procedure StrUpperBuff( S : PAnsiChar)
3509:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3510:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3511:  Procedure StrAddRef( var S : AnsiString)
3512:  Function StrAllocSize( const S : AnsiString) : Longint
3513:  Procedure StrDecRef( var S : AnsiString)
3514:  //Function StrLen( S : PAnsiChar) : Integer
3515:  Function StrLength( const S : AnsiString) : Longint
3516:  Function StrRefCount( const S : AnsiString) : Longint
3517:  Procedure StrResetLength( var S : AnsiString)
3518:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3519:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3520:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3521:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3522:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3523:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3524:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3525:  Function StrFillChar(const C: Char; Count: Integer): string)');
3526:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3527:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3528:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3529:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3530:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3531:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3532:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3533:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3534:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3535:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3536:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3537:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3538:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3539:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3540:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3541:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3542:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3543:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3544:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3545:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3546:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3547:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3548:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3549:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3550:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3551:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3552:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3553:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3554:  Function CharIsBlank( const C : AnsiChar) : Boolean
3555:  Function CharIsControl( const C : AnsiChar) : Boolean
3556:  Function CharIsDelete( const C : AnsiChar) : Boolean
3557:  Function CharIsDigit( const C : AnsiChar) : Boolean
3558:  Function CharIsLower( const C : AnsiChar) : Boolean
3559:  Function CharIsNumberChar( const C : AnsiChar) : Boolean
3560:  Function CharIsPrintable( const C : AnsiChar) : Boolean
3561:  Function CharIsPunctuation( const C : AnsiChar) : Boolean
3562:  Function CharIsReturn( const C : AnsiChar) : Boolean
3563:  Function CharIsSpace( const C : AnsiChar) : Boolean
3564:  Function CharIsUpper( const C : AnsiChar) : Boolean
3565:  Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3566:  Function CharType( const C : AnsiChar) : Word
3567:  Function CharHex( const C : AnsiChar) : Byte
3568:  Function CharLower( const C : AnsiChar) : AnsiChar
3569:  Function CharUpper( const C : AnsiChar) : AnsiChar
3570:  Function CharToggleCase( const C : AnsiChar) : AnsiChar
3571:  Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3572:  Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3573:  Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3574:  Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3575:  Procedure StrIToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3576:  Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3577:  Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3578:  Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3579:  Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3580:  Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
```

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

```
3664: function FirstDayOfPrevMonth: TDateTime;
3665: function LastDayOfPrevMonth: TDateTime;
3666: function FirstDayOfNextMonth: TDateTime;
3667: function ExtractDay(ADate: TDateTime): Word;
3668: function ExtractMonth(ADate: TDateTime): Word;
3669: function ExtractYear(ADate: TDateTime): Word;
3670: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3671: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3672: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3673: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3674: function ValidDate(ADate: TDateTime): Boolean;
3675: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3676: function MonthsBetween(Date1, Date2: TDateTime): Double;
3677: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3678: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3679: function DaysBetween(Date1, Date2: TDateTime): Longint;
3680: { The same as previous but if Date2 < Date1 result = 0 }
3681: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3682: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3683: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3684: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3685: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3686: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3687: { String to date conversions }
3688: function GetDateOrder(const DateFormat: string): TDateOrder;
3689: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3690: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3691: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3692: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3693: function DefDateFormat(FourDigitYear: Boolean): string;
3694: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3695: ------------------------------------------------------------------------------
3696: ******************************** JvUtils;*****************************
3697: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3698: function GetWordOnPos(const S: string; const P: Integer): string;
3699: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3700: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3701: { SubStr returns substring from string, S, separated with Separator string}
3702: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3703: { SubStrEnd same to previous function but Index numerated from the end of string }
3704: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3705: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3706: function SubWord(P: PChar; var P2: PChar): string;
3707: { NumberByWord returns the text representation of
3708:   the number, N, in normal russian language. Was typed from Monitor magazine }
3709: function NumberByWord(const N: Longint): string;
3710: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3711:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3712: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3713: { GetXYByPos is same to previous function, but returns X position in line too}
3714: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3715: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3716: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3717: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3718: function ConcatSep(const S, S2, Separator: string): string;
3719: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3720: function ConcatLeftSep(const S, S2, Separator: string): string;
3721: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3722: function MinimizeString(const S: string; const MaxLen: Integer): string;
3723: { Next 4 function for russian chars transliterating.
3724:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3725: procedure Dos2Win(var S: string);
3726: procedure Win2Dos(var S: string);
3727: function Dos2WinRes(const S: string): string;
3728: function Win2DosRes(const S: string): string;
3729: function Win2Koi(const S: string): string;
3730: { Spaces returns string consists on N space chars }
3731: function Spaces(const N: Integer): string;
3732: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3733: function AddSpaces(const S: string; const N: Integer): string;
3734: { function LastDate for russian users only } { returns date relative to current date: '' }
3735: function LastDate(const Dat: TDateTime): string;
3736: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3737: function CurrencyToStr(const Cur: currency): string;
3738: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3739: function Cmp(const S1, S2: string): Boolean;
3740: { StringCat add S2 string to S1 and returns this string }
3741: function StringCat(var S1: string; S2: string): string;
3742: { HasChar returns True, if Char, Ch, contains in string, S }
3743: function HasChar(const Ch: Char; const S: string): Boolean;
3744: function HasAnyChar(const Chars: string; const S: string): Boolean;
3745: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3746: function CountOfChar(const Ch: Char; const S: string): Integer;
3747: function DefStr(const S: string; Default: string): string;
3748: {**** files routines}
3749: { GetWinDir returns Windows folder name }
3750: function GetWinDir: TFileName;
3751: function GetSysDir: String;
3752: { GetTempDir returns Windows temporary folder name }
```

```
3753: function GetTempDir: string;
3754: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3755: function GenTempFileName(FileName: string): string;
3756: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3757: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3758: { ClearDir clears folder Dir }
3759: function ClearDir(const Dir: string): Boolean;
3760: { DeleteDir clears and than delete folder Dir }
3761: function DeleteDir(const Dir: string): Boolean;
3762: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3763: function FileEquMask(FileName, Mask: TFileName): Boolean;
3764: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3765:   Masks must be separated with comma (';') }
3766: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3767: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3768: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3769: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3770: { FileGetInfo fills SearchRec record for specified file attributes}
3771: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3772: { HasSubFolder returns True, if folder APath contains other folders }
3773: function HasSubFolder(APath: TFileName): Boolean;
3774: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3775: function IsEmptyFolder(APath: TFileName): Boolean;
3776: { AddSlash add slash Char to Dir parameter, if needed }
3777: procedure AddSlash(var Dir: TFileName);
3778: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3779: function AddSlash2(const Dir: TFileName): string;
3780: { AddPath returns FileName with Path, if FileName not contain any path }
3781: function AddPath(const FileName, Path: TFileName): TFileName;
3782: function AddPaths(const PathList, Path: string): string;
3783: function ParentPath(const Path: TFileName): TFileName;
3784: function FindInPath(const FileName, PathList: string): TFileName;
3785: function FindInPaths(const fileName,paths: String): String;
3786: {$IFNDEF BCB1}
3787: { BrowseForFolder displays Browse For Folder dialog }
3788: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3789: {$ENDIF BCB1}
3790: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3791: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3792: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3793: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3794:
3795: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3796: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3797: { HasParam returns True, if program running with specified parameter, Param }
3798: function HasParam(const Param: string): Boolean;
3799: function HasSwitch(const Param: string): Boolean;
3800: function Switch(const Param: string): string;
3801: { ExePath returns ExtractFilePath(ParamStr(0)) }
3802: function ExePath: TFileName;
3803: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3804: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3805: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3806: {**** Graphic routines }
3807: { TTFontSelected returns True, if True Type font is selected in specified device context }
3808: function TTFontSelected(const DC: HDC): Boolean;
3809: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3810: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3811: {**** Windows routines }
3812: { SetWindowTop put window to top without recreating window }
3813: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3814: {**** other routines }
3815: { KeyPressed returns True, if Key VK is now pressed }
3816: function KeyPressed(VK: Integer): Boolean;
3817: procedure SwapInt(var Int1, Int2: Integer);
3818: function IntPower(Base, Exponent: Integer): Integer;
3819: function ChangeTopException(E: TObject): TObject;
3820: function StrToBool(const S: string): Boolean;
3821: {$IFNDEF COMPILER3_UP}
3822: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3823:   Length of MaxLen bytes. The compare operation is controlled by the
3824:   current Windows locale. The return value is the same as for CompareStr. }
3825: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3826: function AnsiStrIComp(S1, S2: PChar): Integer;
3827: {$ENDIF}
3828: function Var2Type(V: Variant; const VarType: Integer): Variant;
3829: function VarToInt(V: Variant): Integer;
3830: function VarToFloat(V: Variant): Double;
3831: { following functions are not documented because they are don't work properly , so don't use them }
3832: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3833: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3834: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3835: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3836: function GetParameter: string;
3837: function GetLongFileName(FileName: string): string;
3838: {* from  FileCtrl}
3839: function DirectoryExists(const Name: string): Boolean;
```

```
3840: procedure ForceDirectories(Dir: string);
3841: {# from  FileCtrl}
3842: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3843: function GetComputerID: string;
3844: function GetComputerName: string;
3845: {**** string routines }
3846: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 function }
3847: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3848: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3849:   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 }
3850: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3851: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3852: function CountOfLines(const S: string): Integer;
3853: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3854: procedure DeleteEmptyLines(Ss: TStrings);
3855: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3856:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3857: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3858: {**** files routines - }
3859: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3860:   Resource can be compressed using MS Compress program}
3861: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3862: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3863: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3864: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3865: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3866: { IniReadSection read section, Section, from ini-file,
3867:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3868:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3869: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3870: { LoadTextFile load text file, FileName, into string }
3871: function LoadTextFile(const FileName: TFileName): string;
3872: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3873: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3874: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3875: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3876: {$IFDEF COMPILER3_UP}
3877: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3878: function TargetFileName(const FileName: TFileName): TFileName;
3879: { return filename ShortCut linked to }
3880: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3881: {$ENDIF COMPILER3_UP}
3882: {**** Graphic routines - }
3883: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3884: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3885: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3886: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3887: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3888: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3889: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3890: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3891: { Cinema draws some visual effect }
3892: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3893: { Roughed fills rect with special 3D pattern }
3894: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3895: { 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 }
3896: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3897: { TextWidth calculate text with for writing using standard desktop font }
3898: function TextWidth(AStr: string): Integer;
3899: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3900: function DefineCursor(Identifer: PChar): TCursor;
3901: {**** other routines - }
3902: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3903: function FindFormByClass(FormClass: TFormClass): TForm;
3904: function FindFormByClassName(FormClassName: string): TForm;
3905: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3906:   having Tag property value, equaled to Tag parameter }
3907: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3908: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3909: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3910: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3911: function RBTag(Parent: TWinControl): Integer;
3912: { AppMinimized returns True, if Application is minimized }
3913: function AppMinimized: Boolean;
3914: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3915:   if Caption parameter = '', it replaced with Application.Title }
3916: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3917: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3918:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3919: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3920:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3921: { Delay stop program execution to MSec msec }
3922: procedure Delay(MSec: Longword);
3923: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3924: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3925: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
```

```
3926: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3927: function PanelBorder(Panel: TCustomPanel): Integer;
3928: function Pixels(Control: TControl; APixels: Integer): Integer;
3929: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3930: procedure Error(const Msg: string);
3931: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3932:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3933:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3934: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3935:   const HideSelColor: Boolean): string;
3936: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3937:   const HideSelColor: Boolean): Integer;
3938: function ItemHtPlain(const Text: string): string;
3939: { ClearList - clears list of TObject }
3940: procedure ClearList(List: TList);
3941: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3942: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3943: { RTTI support }
3944: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3945: function GetPropStr(Obj: TObject; const PropName: string): string;
3946: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3947: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3948: procedure PrepareIniSection(SS: TStrings);
3949: { following functions are not documented because they are don't work properly, so don't use them }
3950: {$IFDEF COMPILER2}
3951: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3952: {$ENDIF}
3953:
3954: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3955: begin
3956:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3957:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3958:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
     Accept:Bool;Sorted:Bool;
3959:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3960:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3961:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3962:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3963:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3964: end;
3965:
3966: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3967: begin
3968:  Const('MaxInitStrNum','LongInt'( 9);
3969: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
     : array of AnsiString; MaxSplit : Integer) : Integer
3970: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
     string; MaxSplit : Integer) : Integer
3971: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
     QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3972: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3973: Function JvStrStrip( S : string) : string
3974: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3975: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3976: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3977: Function StrEatWhiteSpace( const S : string) : string
3978: Function HexToAscii( const S : AnsiString) : AnsiString
3979: Function AsciiToHex( const S : AnsiString) : AnsiString
3980: Function StripQuotes( const S1 : AnsiString) : AnsiString
3981: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3982: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3983: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3984: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3985: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3986: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3987: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3988: Function JvValidIdentifier( S1 : String) : Boolean
3989: Function JvEndChar( X : AnsiChar) : Boolean
3990: Procedure JvGetToken( S1, S2 : PAnsiChar)
3991: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
3992: Function IsKeyword( S1 : PAnsiChar) : Boolean
3993: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
3994: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
3995: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3996: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3997: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3998: Function GetTokenCount : Integer
3999: Procedure ResetTokenCount
4000: end;
4001:
4002: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
4003: begin
4004:   SIRegister_TJvQueryParamsDialog(CL);
4005:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
4006: end;
4007:
4008: ******************************** JvStrUtil /  JvStrUtils;****************************
4009: function FindNotBlankCharPos(const S: string): Integer;
4010: function AnsiChangeCase(const S: string): string;
```

```
4011: function GetWordOnPos(const S: string; const P: Integer): string;
4012: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4013: function Cmp(const S1, S2: string): Boolean;
4014: { Spaces returns string consists on N space chars }
4015: function Spaces(const N: Integer): string;
4016: { HasChar returns True, if char, Ch, contains in string, S }
4017: function HasChar(const Ch: Char; const S: string): Boolean;
4018: function HasAnyChar(const Chars: string; const S: string): Boolean;
4019: { SubStr returns substring from string, S, separated with Separator string}
4020: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4021: { SubStrEnd same to previous function but Index numerated from the end of string }
4022: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4023: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4024: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
4025: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4026: { GetXYByPos is same to previous function, but returns X position in line too}
4027: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4028: { AddSlash returns string with added slash char to Dir parameter, if needed }
4029: function AddSlash2(const Dir: TFileName): string;
4030: { AddPath returns FileName with Path, if FileName not contain any path }
4031: function AddPath(const FileName, Path: TFileName): TFileName;
4032: { ExePath returns ExtractFilePath(ParamStr(0)) }
4033: function ExePath: TFileName;
4034: function LoadTextFile(const FileName: TFileName): string;
4035: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4036: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4037: function ConcatSep(const S, S2, Separator: string): string;
4038: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4039: function FileEquMask(FileName, Mask: TFileName): Boolean;
4040: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4041:   Masks must be separated with comma (';') }
4042: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4043: function StringEndsWith(const Str, SubStr: string): Boolean;
4044: function ExtractFilePath2(const FileName: string): string;
4045: function StrToOem(const AnsiStr: string): string;
4046: { StrToOem translates a string from the Windows character set into the OEM character set. }
4047: function OemToAnsiStr(const OemStr: string): string;
4048: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4049: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4050: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4051: function ReplaceStr(const S, Srch, Replace: string): string;
4052: { Returns string with every occurrence of Srch string replaced with Replace string. }
4053: function DelSpace(const S: string): string;
4054: { DelSpace return a string with all white spaces removed. }
4055: function DelChars(const S: string; Chr: Char): string;
4056: { DelChars return a string with all Chr characters removed. }
4057: function DelBSpace(const S: string): string;
4058: { DelBSpace trims leading spaces from the given string. }
4059: function DelESpace(const S: string): string;
4060: { DelESpace trims trailing spaces from the given string. }
4061: function DelRSpace(const S: string): string;
4062: { DelRSpace trims leading and trailing spaces from the given string. }
4063: function DelSpace1(const S: string): string;
4064: { DelSpace1 return a string with all non-single white spaces removed. }
4065: function Tab2Space(const S: string; Numb: Byte): string;
4066: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4067: function NPos(const C: string; S: string; N: Integer): Integer;
4068: { NPos searches for a N-th position of substring C in a given string. }
4069: function MakeStr(C: Char; N: Integer): string;
4070: function MS(C: Char; N: Integer): string;
4071: { MakeStr return a string of length N filled with character C. }
4072: function AddChar(C: Char; const S: string; N: Integer): string;
4073: { AddChar return a string left-padded to length N with characters C. }
4074: function AddCharR(C: Char; const S: string; N: Integer): string;
4075: { AddCharR return a string right-padded to length N with characters C. }
4076: function LeftStr(const S: string; N: Integer): string;
4077: { LeftStr return a string right-padded to length N with blanks. }
4078: function RightStr(const S: string; N: Integer): string;
4079: { RightStr return a string left-padded to length N with blanks. }
4080: function CenterStr(const S: string; Len: Integer): string;
4081: { CenterStr centers the characters in the string based upon the Len specified. }
4082: function CompStr(const S1, S2: string): Integer;
4083: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4084: function CompText(const S1, S2: string): Integer;
4085: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4086: function Copy2Symb(const S: string; Symb: Char): string;
4087: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4088: function Copy2SymbDel(var S: string; Symb: Char): string;
4089: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4090: function Copy2Space(const S: string): string;
4091: { Copy2Symb returns a substring of a string S from begining to first white space. }
4092: function Copy2SpaceDel(var S: string): string;
4093: { Copy2SpaceDel returns a substring of a string S from begining to first
4094:   white space and removes this substring from S. }
4095: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4096: { Returns string, with the first letter of each word in uppercase,
4097:   all other letters in lowercase. Words are delimited by WordDelims. }
4098: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4099: { WordCount given a set of word delimiters, returns number of words in S. }
```

```
4100: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4101: { Given a set of word delimiters, returns start position of N'th word in S. }
4102: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4103: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4104: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4105: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4106:   delimiters, return the N'th word in S. }
4107: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4108: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4109: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4110: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4111: function QuotedString(const S: string; Quote: Char): string;
4112: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4113: function ExtractQuotedString(const S: string; Quote: Char): string;
4114: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4115:    and reduces pairs of Quote characters within the quoted string to a single character. }
4116: function FindPart(const HelpWilds, InputStr: string): Integer;
4117: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4118: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4119: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4120: function XorString(const Key, Src: ShortString): ShortString;
4121: function XorEncode(const Key, Source: string): string;
4122: function XorDecode(const Key, Source: string): string;
4123: { ** Command line routines ** }
4124: {$IFNDEF COMPILER4_UP}
4125: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4126: {$ENDIF}
4127: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4128: { ** Numeric string handling routines ** }
4129: function Numb2USA(const S: string): string;
4130: { Numb2USA converts numeric string S to USA-format. }
4131: function Dec2Hex(N: Longint; A: Byte): string;
4132: function D2H(N: Longint; A: Byte): string;
4133: { Dec2Hex converts the given value to a hexadecimal string representation
4134:   with the minimum number of digits (A) specified. }
4135: function Hex2Dec(const S: string): Longint;
4136: function H2D(const S: string): Longint;
4137: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4138: function Dec2Numb(N: Longint; A, B: Byte): string;
4139: { Dec2Numb converts the given value to a string representation with the
4140:   base equal to B and with the minimum number of digits (A) specified. }
4141: function Numb2Dec(S: string; B: Byte): Longint;
4142: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4143: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4144: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4145: function IntToRoman(Value: Longint): string;
4146: { IntToRoman converts the given value to a roman numeric string representation. }
4147: function RomanToInt(const S: string): Longint;
4148: { RomanToInt converts the given string to an integer value. If the string
4149:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4150: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4151: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4152: ********************************* JvFileUtil;*********************************
4153: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4154: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4155: procedure MoveFile(const FileName, DestName: TFileName);
4156: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4157: {$IFDEF COMPILER4_UP}
4158: function GetFileSize(const FileName: string): Int64;
4159: {$ELSE}
4160: function GetFileSize(const FileName: string): Longint;
4161: {$ENDIF}
4162: function FileDateTime(const FileName: string): TDateTime;
4163: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4164: function DeleteFiles(const FileMask: string): Boolean;
4165: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4166: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4167: function NormalDir(const DirName: string): string;
4168: function RemoveBackSlash(const DirName: string): string;
4169: function ValidFileName(const FileName: string): Boolean;
4170: function DirExists(Name: string): Boolean;
4171: procedure ForceDirectories(Dir: string);
4172: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4173:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4174: {$IFDEF COMPILER4_UP}
4175: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4176: {$ENDIF}
4177: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4178: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4179: {$IFDEF COMPILER4_UP}
4180: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4181: {$ENDIF}
4182: function GetTempDir: string;
4183: function GetWindowsDir: string;
4184: function GetSystemDir: string;
4185: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4186: {$IFDEF WIN32}
4187: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
```

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

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

```
4364: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4365: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4366: Procedure DisposeMem( var Buffer, Size : Integer)
4367: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4368: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4369: Function VarTypeToDataType( VarType : Integer) : TFieldType
4370: ******************************************unit JvStrings;*************************************
4371: {template functions}
4372: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4373: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4374: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4375: function RemoveMasterBlocks(const SourceStr: string): string;
4376: function RemoveFields(const SourceStr: string): string;
4377: {http functions}
4378: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4379: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4380: {set functions}
4381: procedure SplitSet(AText: string; AList: TStringList);
4382: function JoinSet(AList: TStringList): string;
4383: function FirstOfSet(const AText: string): string;
4384: function LastOfSet(const AText: string): string;
4385: function CountOfSet(const AText: string): Integer;
4386: function SetRotateRight(const AText: string): string;
4387: function SetRotateLeft(const AText: string): string;
4388: function SetPick(const AText: string; AIndex: Integer): string;
4389: function SetSort(const AText: string): string;
4390: function SetUnion(const Set1, Set2: string): string;
4391: function SetIntersect(const Set1, Set2: string): string;
4392: function SetExclude(const Set1, Set2: string): string;
4393: {replace any <,> etc by &lt; &gt;}
4394: function XMLSafe(const AText: string): string;
4395: {simple hash, Result can be used in Encrypt}
4396: function Hash(const AText: string): Integer;
4397: { Base64 encode and decode a string }
4398: function B64Encode(const S: AnsiString): AnsiString;
4399: function B64Decode(const S: AnsiString): AnsiString;
4400: {Basic encryption from a Borland Example}
4401: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4402: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4403: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4404: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4405: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4406: procedure CSVToTags(Src, Dst: TStringList);
4407: // converts a csv list to a tagged string list
4408: procedure TagsToCSV(Src, Dst: TStringList);
4409: // converts a tagged string list to a csv list
4410: // only fieldnames from the first record are scanned ib the other records
4411: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4412: {selects akey=avalue from Src and returns recordset in Dst}
4413: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4414: {filters Src for akey=avalue}
4415: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4416: {orders a tagged Src list by akey}
4417: function PosStr(const FindString, SourceString: string;
4418:   StartPos: Integer = 1): Integer;
4419: { PosStr searches the first occurrence of a substring FindString in a string
4420:   given by SourceString with case sensitivity (upper and lower case characters
4421:   are differed). This function returns the index value of the first character
4422:   of a specified substring from which it occurs in a given string starting with
4423:   StartPos character index. If a specified substring is not found Q_PosStr
4424:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4425: function PosStrLast(const FindString, SourceString: string): Integer;
4426: {finds the last occurance}
4427: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4428: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4429: { PosText searches the first occurrence of a substring FindString in a string
4430:   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
4431: function PosTextLast(const FindString, SourceString: string): Integer;
4432: {finds the last occurance}
4433: function NameValuesToXML(const AText: string): string;
4434: {$IFDEF MSWINDOWS}
4435: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4436: {$ENDIF MSWINDOWS}
4437: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4438: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4439: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4440: procedure SaveString(const AFile, AText: string);
4441: Procedure SaveStringasFile( const AFile, AText : string)
4442: function LoadStringJ(const AFile: string): string;
4443: Function LoadStringofFile( const AFile : string) : string
4444: Procedure SaveStringtoFile( const AFile, AText : string)
4445: Function LoadStringfromFile( const AFile : string) : string
4446: function HexToColor(const AText: string): TColor;
4447: function UppercaseHTMLTags(const AText: string): string;
4448: function LowercaseHTMLTags(const AText: string): string;
4449: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4450: function RelativePath(const ASrc, ADst: string): string;
```

```
4451: function GetToken(var Start: Integer; const SourceText: string): string;
4452: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4453: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4454: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4455: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4456: // parses the beginning of an attribute: space + alpha character
4457: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4458: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4459: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4460: // parses all name=value attributes to the attributes TStringList
4461: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4462: // checks if a name="value" pair exists and returns any value
4463: function GetStrValue(const AText, AName, ADefault: string): string;
4464: // retrieves string value from a line like:
4465: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4466: // returns ADefault when not found
4467: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4468: // same for a color
4469: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4470: // same for an Integer
4471: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4472: // same for a float
4473: function GetBoolValue(const AText, AName: string): Boolean;
4474: // same for Boolean but without default
4475: function GetValue(const AText, AName: string): string;
4476: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4477: procedure SetValue(var AText: string; const AName, AValue: string);
4478: // sets a string value in a line
4479: procedure DeleteValue(var AText: string; const AName: string);
4480: // deletes a AName="value" pair from AText
4481: procedure GetNames(AText: string; AList: TStringList);
4482: // get a list of names from a string with name="value" pairs
4483: function GetHTMLColor(AColor: TColor): string;
4484: // converts a color value to the HTML hex value
4485: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4486: // finds a string backward case sensitive
4487: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4488: // finds a string backward case insensitive
4489: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4490:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4491: // finds a text range, e.g. <TD>....</TD> case sensitive
4492: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4493:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4494: // finds a text range, e.g. <TD>....</td> case insensitive
4495: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4496:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4497: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4498: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4499:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4500: // finds a text range backward, e.g. <TD>....</td> case insensitive
4501: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4502:   var RangeEnd: Integer): Boolean;
4503: // finds a HTML or XML tag:  <....>
4504: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4505:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4506: // finds the innertext between opening and closing tags
4507: function Easter(NYear: Integer): TDateTime;
4508: // returns the easter date of a year.
4509: function GetWeekNumber(Today: TDateTime): string;
4510: //gets a datecode. Returns year and weeknumber in format: YYWW
4511: function ParseNumber(const S: string): Integer;
4512: // parse number returns the last position, starting from 1
4513: function ParseDate(const S: string): Integer;
4514: // parse a SQL style data string from positions 1,
4515: // starts and ends with #
4516:
4517: ****************************************unit JvJCLUtils;****************************************
4518:
4519: function VarIsInt(Value: Variant): Boolean;
4520:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4521: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4522: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4523: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4524: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4525: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4526: function GetWordOnPos(const S: string; const P: Integer): string;
4527: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4528: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4529: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4530: { GetWordOnPosEx working like GetWordOnPos function, but
4531:   also returns Word position in iBeg, iEnd variables }
4532: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4533: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4534: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4535: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4536: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4537: { GetEndPosCaret returns the caret position of the last char. For the position
4538:   after the last char of Text you must add 1 to the returned X value. }
4539: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
```

```
4540: { GetEndPosCaret returns the caret position of the last char. For the position
4541:   after the last char of Text you must add 1 to the returned X value. }
4542: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4543: function SubStrBySeparator(const S:string;const Index:Integer;const
      Separator:string;StartIndex:Int=1):string;
4544: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
      Separator:WideString;StartIndex:Int:WideString;
4545: { SubStrEnd same to previous function but Index numerated from the end of string }
4546: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4547: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4548: function SubWord(P: PChar; var P2: PChar): string;
4549: function CurrencyByWord(Value: Currency): string;
4550: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4551: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4552: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4553: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4554: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4555: { ReplaceString searches for all substrings, OldPattern,
4556:   in a string, S, and replaces them with NewPattern }
4557: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4558: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
      WideString;StartIndex:Integer=1):WideString;
4559: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4560: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4561: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4562: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4563:
4564: { Next 4 function for russian chars transliterating.
4565:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4566: procedure Dos2Win(var S: AnsiString);
4567: procedure Win2Dos(var S: AnsiString);
4568: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4569: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4570: function Win2Koi(const S: AnsiString): AnsiString;
4571: { FillString fills the string Buffer with Count Chars }
4572: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4573: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4574: { MoveString copies Count Chars from Source to Dest }
4575: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4576: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4577:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4578: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4579: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4580: { MoveWideChar copies Count WideChars from Source to Dest }
4581: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4582: { FillNativeChar fills Buffer with Count NativeChars }
4583: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4584: { MoveWideChar copies Count WideChars from Source to Dest }
4585: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4586: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4587: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4588: { Spaces returns string consists on N space chars }
4589: function Spaces(const N: Integer): string;
4590: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4591: function AddSpaces(const S: string; const N: Integer): string;
4592: function SpacesW(const N: Integer): WideString;
4593: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4594: { function LastDateRUS for russian users only }
4595: { returns date relative to current date: 'äâà äíÿ íàçàä' }
4596: function LastDateRUS(const Dat: TDateTime): string;
4597: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4598: function CurrencyToStr(const Cur: Currency): string;
4599: { HasChar returns True, if Char, Ch, contains in string, S }
4600: function HasChar(const Ch: Char; const S: string): Boolean;
4601: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4602: function HasAnyChar(const Chars: string; const S: string): Boolean;
4603: {$IFNDEF COMPILER12_UP}
4604: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4605: {$ENDIF ~COMPILER12_UP}
4606: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4607: function CountOfChar(const Ch: Char; const S: string): Integer;
4608: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4609: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4610: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4611: function StrPosW(S, SubStr: PWideChar): PWideChar;
4612: function StrLenW(S: PWideChar): Integer;
4613: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4614: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4615: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4616: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
```

```
4617: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4618:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4619: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4620: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4621: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4622: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4623: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4624: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4625: Function ScreenPixelFormat : TPixelFormat
4626: Function ScreenColorCount : Integer
4627: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4628: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4629: //  SIRegister_TJvGradient(CL);
4630:
4631: {************************************* files routines}
4632: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4633: const
4634:   {$IFDEF MSWINDOWS}
4635:   DefaultCaseSensitivity = False;
4636:   {$ENDIF MSWINDOWS}
4637:   {$IFDEF UNIX}
4638:   DefaultCaseSensitivity = True;
4639:   {$ENDIF UNIX}
4640: { GenTempFileName returns temporary file name on
4641:   drive, there FileName is placed }
4642: function GenTempFileName(FileName: string): string;
4643: { GenTempFileNameExt same to previous function, but
4644:   returning filename has given extension, FileExt }
4645: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4646: { ClearDir clears folder Dir }
4647: function ClearDir(const Dir: string): Boolean;
4648: { DeleteDir clears and than delete folder Dir }
4649: function DeleteDir(const Dir: string): Boolean;
4650: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4651: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4652: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4653:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4654: function FileEquMasks(FileName, Masks: TFileName;
4655:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4656: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4657: {$IFDEF MSWINDOWS}
4658: { LZFileExpand expand file, FileSource,
4659:   into FileDest. Given file must be compressed, using MS Compress program }
4660: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4661: {$ENDIF MSWINDOWS}
4662: { FileGetInfo fills SearchRec record for specified file attributes}
4663: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4664: { HasSubFolder returns True, if folder APath contains other folders }
4665: function HasSubFolder(APath: TFileName): Boolean;
4666: { IsEmptyFolder returns True, if there are no files or
4667:   folders in given folder, APath}
4668: function IsEmptyFolder(APath: TFileName): Boolean;
4669: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4670: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4671: { AddPath returns FileName with Path, if FileName not contain any path }
4672: function AddPath(const FileName, Path: TFileName): TFileName;
4673: function AddPaths(const PathList, Path: string): string;
4674: function ParentPath(const Path: TFileName): TFileName;
4675: function FindInPath(const FileName, PathList: string): TFileName;
4676: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4677: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4678: { HasParam returns True, if program running with specified parameter, Param }
4679: function HasParam(const Param: string): Boolean;
4680: function HasSwitch(const Param: string): Boolean;
4681: function Switch(const Param: string): string;
4682: { ExePath returns ExtractFilePath(ParamStr(0)) }
4683: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4684: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4685: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4686: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4687: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4688: {**** Graphic routines }
4689: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4690: function IsTTFontSelected(const DC: HDC): Boolean;
4691: function KeyPressed(VK: Integer): Boolean;
4692: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4693: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4694: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4695: {**** Color routines }
4696: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4697: function RGBToBGR(Value: Cardinal): Cardinal;
4698: //function ColorToPrettyName(Value: TColor): string;
4699: //function PrettyNameToColor(const Value: string): TColor;
4700: {**** other routines }
4701: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4702: function IntPower(Base, Exponent: Integer): Integer;
4703: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4704: function StrToBool(const S: string): Boolean;
4705: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
```

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

```
4792: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4793: // (rom) from JvBandUtils to make it obsolete
4794: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4795: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4796: function CreateIconFromClipboard: TIcon;
4797: { begin JvIconClipboardUtils } { Icon clipboard routines }
4798: function CF_ICON: Word;
4799: procedure AssignClipboardIcon(Icon: TIcon);
4800: { Real-size icons support routines (32-bit only) }
4801: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4802: function CreateRealSizeIcon(Icon: TIcon): HICON;
4803: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4804: {end JvIconClipboardUtils }
4805: function CreateScreenCompatibleDC: HDC;
4806: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF}
4807: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4808: { begin JvRLE } // (rom) changed API for inclusion in JCL
4809: procedure RleCompressTo(InStream, OutStream: TStream);
4810: procedure RleDecompressTo(InStream, OutStream: TStream);
4811: procedure RleCompress(Stream: TStream);
4812: procedure RleDecompress(Stream: TStream);
4813: { end JvRLE } { begin JvDateUtil }
4814: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4815: function IsLeapYear(AYear: Integer): Boolean;
4816: function DaysInAMonth(const AYear, AMonth: Word): Word;
4817: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4818: function FirstDayOfPrevMonth: TDateTime;
4819: function LastDayOfPrevMonth: TDateTime;
4820: function FirstDayOfNextMonth: TDateTime;
4821: function ExtractDay(ADate: TDateTime): Word;
4822: function ExtractMonth(ADate: TDateTime): Word;
4823: function ExtractYear(ADate: TDateTime): Word;
4824: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4825: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4826: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4827: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4828: function ValidDate(ADate: TDateTime): Boolean;
4829: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4830: function MonthsBetween(Date1, Date2: TDateTime): Double;
4831: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4832: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4833: function DaysBetween(Date1, Date2: TDateTime): Longint;
4834: { The same as previous but if Date2 < Date1 result = 0 }
4835: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4836: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4837: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4838: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4839: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4840: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4841: { String to date conversions }
4842: function GetDateOrder(const DateFormat: string): TDateOrder;
4843: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4844: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4845: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4846: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4847: //function DefDateFormat(AFourDigitYear: Boolean): string;
4848: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4849: function FormatLongDate(Value: TDateTime): string;
4850: function FormatLongDateTime(Value: TDateTime): string;
4851: { end JvDateUtil }
4852: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4853: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4854: { begin JvStrUtils } { ** Common string handling routines ** }
4855: {$IFDEF UNIX}
4856: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4857:   const ToCode, FromCode: AnsiString): Boolean;
4858: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4859: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4860: function OemStrToAnsi(const S: AnsiString): AnsiString;
4861: function AnsiStrToOem(const S: AnsiString): AnsiString;
4862: {$ENDIF UNIX}
4863: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4864: { StrToOem translates a string from the Windows character set into the OEM character set. }
4865: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4866: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4867: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4868: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4869: function ReplaceStr(const S, Srch, Replace: string): string;
4870: { Returns string with every occurrence of Srch string replaced with Replace string. }
4871: function DelSpace(const S: string): string;
4872: { DelSpace return a string with all white spaces removed. }
4873: function DelChars(const S: string; Chr: Char): string;
4874: { DelChars return a string with all Chr characters removed. }
4875: function DelBSpace(const S: string): string;
4876: { DelBSpace trims leading spaces from the given string. }
4877: function DelESpace(const S: string): string;
4878: { DelESpace trims trailing spaces from the given string. }
4879: function DelRSpace(const S: string): string;
```

```
4880: { DelRSpace trims leading and trailing spaces from the given string. }
4881: function DelSpace1(const S: string): string;
4882: { DelSpace1 return a string with all non-single white spaces removed. }
4883: function Tab2Space(const S: string; Numb: Byte): string;
4884: { Tab2Space converts any tabulation character in the given string to the
4885:   Numb spaces characters. }
4886: function NPos(const C: string; S: string; N: Integer): Integer;
4887: { NPos searches for a N-th position of substring C in a given string. }
4888: function MakeStr(C: Char; N: Integer): string; overload;
4889: {$IFNDEF COMPILER12_UP}
4890: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4891: {$ENDIF !COMPILER12_UP}
4892: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4893: { MakeStr return a string of length N filled with character C. }
4894: function AddChar(C: Char; const S: string; N: Integer): string;
4895: { AddChar return a string left-padded to length N with characters C. }
4896: function AddCharR(C: Char; const S: string; N: Integer): string;
4897: { AddCharR return a string right-padded to length N with characters C. }
4898: function LeftStr(const S: string; N: Integer): string;
4899: { LeftStr return a string right-padded to length N with blanks. }
4900: function RightStr(const S: string; N: Integer): string;
4901: { RightStr return a string left-padded to length N with blanks. }
4902: function CenterStr(const S: string; Len: Integer): string;
4903: { CenterStr centers the characters in the string based upon the Len specified. }
4904: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4905: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4906:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4907: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4908: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4909: function Copy2Symb(const S: string; Symb: Char): string;
4910: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4911: function Copy2SymbDel(var S: string; Symb: Char): string;
4912: { Copy2SymbDel returns a substring of a string S from begining to first
4913:   character Symb and removes this substring from S. }
4914: function Copy2Space(const S: string): string;
4915: { Copy2Symb returns a substring of a string S from begining to first white space. }
4916: function Copy2SpaceDel(var S: string): string;
4917: { Copy2SpaceDel returns a substring of a string S from begining to first
4918:   white space and removes this substring from S. }
4919: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4920: { Returns string, with the first letter of each word in uppercase,
4921:   all other letters in lowercase. Words are delimited by WordDelims. }
4922: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4923: { WordCount given a set of word delimiters, returns number of words in S. }
4924: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4925: { Given a set of word delimiters, returns start position of N'th word in S. }
4926: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4927: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4928: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4929: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4930:   delimiters, return the N'th word in S. }
4931: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4932: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4933:   that started from position Pos. }
4934: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4935: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4936: function QuotedString(const S: string; Quote: Char): string;
4937: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4938: function ExtractQuotedString(const S: string; Quote: Char): string;
4939: { ExtractQuotedString removes the Quote characters from the beginning and
4940:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4941: function FindPart(const HelpWilds, InputStr: string): Integer;
4942: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4943: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4944: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4945: function XorString(const Key, Src: ShortString): ShortString;
4946: function XorEncode(const Key, Source: string): string;
4947: function XorDecode(const Key, Source: string): string;
4948: { ** Command line routines ** }
4949: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4950: { ** Numeric string handling routines ** }
4951: function Numb2USA(const S: string): string;
4952: { Numb2USA converts numeric string S to USA-format. }
4953: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4954: { Dec2Hex converts the given value to a hexadecimal string representation
4955:   with the minimum number of digits (A) specified. }
4956: function Hex2Dec(const S: string): Longint;
4957: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4958: function Dec2Numb(N: Int64; A, B: Byte): string;
4959: { Dec2Numb converts the given value to a string representation with the
4960:   base equal to B and with the minimum number of digits (A) specified. }
4961: function Numb2Dec(S: string; B: Byte): Int64;
4962: { Numb2Dec converts the given B-based numeric string to the corresponding
4963:   integer value. }
4964: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4965: { IntToBin converts the given value to a binary string representation
4966:   with the minimum number of digits specified. }
4967: function IntToRoman(Value: Longint): string;
4968: { IntToRoman converts the given value to a roman numeric string representation. }
```

```
4969: function RomanToInt(const S: string): Longint;
4970: { RomanToInt converts the given string to an integer value. If the string
4971:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4972: function FindNotBlankCharPos(const S: string): Integer;
4973: function FindNotBlankCharPosW(const S: WideString): Integer;
4974: function AnsiChangeCase(const S: string): string;
4975: function WideChangeCase(const S: string): string;
4976: function StartsText(const SubStr, S: string): Boolean;
4977: function EndsText(const SubStr, S: string): Boolean;
4978: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4979: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4980: {end JvStrUtils}
4981: {$IFDEF UNIX}
4982: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4983: {$ENDIF UNIX}
4984: { begin JvFileUtil }
4985: function FileDateTime(const FileName: string): TDateTime;
4986: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4987: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4988: function NormalDir(const DirName: string): string;
4989: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4990: function ValidFileName(const FileName: string): Boolean;
4991: {$IFDEF MSWINDOWS}
4992: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4993: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4994: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4995: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4996: {$ENDIF MSWINDOWS}
4997: function GetWindowsDir: string;
4998: function GetSystemDir: string;
4999: function ShortToLongFileName(const ShortName: string): string;
5000: function LongToShortFileName(const LongName: string): string;
5001: function ShortToLongPath(const ShortName: string): string;
5002: function LongToShortPath(const LongName: string): string;
5003: {$IFDEF MSWINDOWS}
5004: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
5005: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5006: {$ENDIF MSWINDOWS}
5007: { end JvFileUtil }
5008: // Works like PtInRect but includes all edges in comparision
5009: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5010: // Works like PtInRect but excludes all edges from comparision
5011: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5012: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5013: function IsFourDigitYear: Boolean;
5014: { moved from JvJVCLUTils }
5015: //Open an object with the shell (url or something like that)
5016: function OpenObject(const Value: string): Boolean; overload;
5017: function OpenObject(Value: PChar): Boolean; overload;
5018: {$IFDEF MSWINDOWS}
5019: //Raise the last Exception
5020: procedure RaiseLastWin32; overload;
5021: procedure RaiseLastWin32(const Text: string); overload;
5022: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
5023: function GetFileVersion(const AFileName: string): Cardinal;
5024: {$EXTERNALSYM GetFileVersion}
5025: //Get version of Shell.dll
5026: function GetShellVersion: Cardinal;
5027: {$EXTERNALSYM GetShellVersion}
5028: // CD functions  on HW
5029: procedure OpenCdDrive;
5030: procedure CloseCdDrive;
5031: // returns True if Drive is accessible
5032: function DiskInDrive(Drive: Char): Boolean;
5033: {$ENDIF MSWINDOWS}
5034: //Same as linux function ;)
5035: procedure PError(const Text: string);
5036: // execute a program without waiting
5037: procedure Exec(const FileName, Parameters, Directory: string);
5038: // execute a program and wait for it to finish
5039: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5040: // returns True if this is the first instance of the program that is running
5041: function FirstInstance(const ATitle: string): Boolean;
5042: // restores a window based on it's classname and Caption. Either can be left empty
5043: // to widen the search
5044: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5045: // manipulate the traybar and start button
5046: procedure HideTraybar;
5047: procedure ShowTraybar;
5048: procedure ShowStartButton(Visible: Boolean = True);
5049: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5050: procedure MonitorOn;
5051: procedure MonitorOff;
5052: procedure LowPower;
5053: // send a key to the window named AppName
5054: function SendKey(const AppName: string; Key: Char): Boolean;
5055: {$IFDEF MSWINDOWS}
```

```
5056: // returns a list of all win currently visible, the Objects property is filled with their window handle
5057: procedure GetVisibleWindows(List: TStrings);
5058: Function GetVisibleWindowsF( List : TStrings):TStrings');
5059: // associates an extension to a specific program
5060: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5061: procedure AddToRecentDocs(const FileName: string);
5062: function GetRecentDocs: TStringList;
5063: {$ENDIF MSWINDOWS}
5064: function CharIsMoney(const Ch: Char): Boolean;
5065: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5066: function IntToExtended(I: Integer): Extended;
5067: { GetChangedText works out the new text given the current cursor pos & the key pressed
5068:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5069: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5070: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5071: //function StrIsInteger(const S: string): Boolean;
5072: function StrIsFloatMoney(const Ps: string): Boolean;
5073: function StrIsDateTime(const Ps: string): Boolean;
5074: function PreformatDateString(Ps: string): string;
5075: function BooleanToInteger(const B: Boolean): Integer;
5076: function StringToBoolean(const Ps: string): Boolean;
5077: function SafeStrToDateTime(const Ps: string): TDateTime;
5078: function SafeStrToDate(const Ps: string): TDateTime;
5079: function SafeStrToTime(const Ps: string): TDateTime;
5080: function StrDelete(const psSub, psMain: string): string;
5081:   { returns the fractional value of pcValue}
5082: function TimeOnly(pcValue: TDateTime): TTime;
5083: { returns the integral value of pcValue }
5084: function DateOnly(pcValue: TDateTime): TDate;
5085: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5086: const { TDateTime value used to signify Null value}
5087:   NullEquivalentDate: TDateTime = 0.0;
5088: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5089: // Replacement for Win32Check to avoid platform specific warnings in D6
5090: function OSCheck(RetVal: Boolean): Boolean;
5091: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5092:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5093:   not be forced to use FileCtrl unnecessarily }
5094: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5095: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5096: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5097: function MinimizeString(const S: string; const MaxLen: Integer): string;
5098: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
5099: { 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.}
5100: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5101: {$ENDIF MSWINDOWS}
5102: procedure ResourceNotFound(ResID: PChar);
5103: function EmptyRect: TRect;
5104: function RectWidth(R: TRect): Integer;
5105: function RectHeight(R: TRect): Integer;
5106: function CompareRect(const R1, R2: TRect): Boolean;
5107: procedure RectNormalize(var R: TRect);
5108: function RectIsSquare(const R: TRect): Boolean;
5109: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5110: //If AMaxSize = -1 ,then auto calc Square's max size
5111: {$IFDEF MSWINDOWS}
5112: procedure FreeUnusedOle;
5113: function GetWindowsVersion: string;
5114: function LoadDLL(const LibName: string): THandle;
5115: function RegisterServer(const ModuleName: string): Boolean;
5116: function UnregisterServer(const ModuleName: string): Boolean;
5117: {$ENDIF MSWINDOWS}
5118: { String routines }
5119: function GetEnvVar(const VarName: string): string;
5120: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5121: function StringToPChar(var S: string): PChar;
5122: function StrPAlloc(const S: string): PChar;
5123: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5124: function DropT(const S: string): string;
5125: { Memory routines }
5126: function AllocMemo(Size: Longint): Pointer;
5127: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5128: procedure FreeMemo(var fpBlock: Pointer);
5129: function GetMemoSize(fpBlock: Pointer): Longint;
5130: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5131: { Manipulate huge pointers routines }
5132: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5133: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5134: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5135: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5136: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5137: function WindowClassName(Wnd: THandle): string;
5138: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5139: procedure ActivateWindow(Wnd: THandle);
5140: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5141: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
```

```
5142: { SetWindowTop put window to top without recreating window }
5143: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5144: procedure CenterWindow(Wnd: THandle);
5145: function MakeVariant(const Values: array of Variant): Variant;
5146: { Convert dialog units to pixels and backwards }
5147: {$IFDEF MSWINDOWS}
5148: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5149: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5150: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5151: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5152: {$ENDIF MSWINDOWS}
5153: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5154: {$IFDEF BCB}
5155: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5156: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5157: {$ELSE}
5158: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5159: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5160: {$ENDIF BCB}
5161: {$IFDEF MSWINDOWS}
5162: { BrowseForFolderNative displays Browse For Folder dialog }
5163: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5164: {$ENDIF MSWINDOWS}
5165: procedure AntiAlias(Clip: TBitmap);
5166: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5167: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5168:   ABitmap: TBitmap; const SourceRect: TRect);
5169: function IsTrueType(const FontName: string): Boolean;
5170: // Removes all non-numeric characters from AValue and returns the resulting string
5171: function TextToValText(const AValue: string): string;
5172: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5173: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5174: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5175: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5176: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5177:
5178: ****************************************************unit uPSI_JvTFUtils;
5179:  Function JExtractYear( ADate : TDateTime) : Word
5180:  Function JExtractMonth( ADate : TDateTime) : Word
5181:  Function JExtractDay( ADate : TDateTime) : Word
5182:  Function ExtractHours( ATime : TDateTime) : Word
5183:  Function ExtractMins( ATime : TDateTime) : Word
5184:  Function ExtractSecs( ATime : TDateTime) : Word
5185:  Function ExtractMSecs( ATime : TDateTime) : Word
5186:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5187:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5188:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5189:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5190:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5191:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5192:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5193:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5194:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5195:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5196:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5197:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5198:  Procedure EnsureMonth( Month : Word)
5199:  Procedure EnsureDOW( DOW : Word)
5200:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5201:  Function Lesser( N1, N2 : Integer) : Integer
5202:  Function Greater( N1, N2 : Integer) : Integer
5203:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5204:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5205:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5206:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5207:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5208:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5209:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
     AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5210:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
     HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5211:  Function JRectWidth( ARect : TRect) : Integer
5212:  Function JRectHeight( ARect : TRect) : Integer
5213:  Function JEmptyRect : TRect
5214:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5215:
5216: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5217: begin
5218:  Procedure HideTaskBarButton( hWindow : HWND)
5219:  Function msLoadStr( ID : Integer) : String
5220:  Function msFormat( fmt : String; params : array of const) : String
5221:  Function msFileExists( const FileName : String) : Boolean
5222:  Function msIntToStr( Int : Int64) : String
5223:  Function msStrPas( const Str : PChar) : String
5224:  Function msRenameFile( const OldName, NewName : String) : Boolean
5225:  Function CutFileName( s : String) : String
5226:  Function GetVersionInfo( var VersionString : String) : DWORD
5227:  Function FormatTime( t : Cardinal) : String
```

```
5228:  Function msCreateDir( const Dir : string) : Boolean
5229:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5230:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5231:  Function msStrLen( Str : PChar) : Integer
5232:  Function msDirectoryExists( const Directory : String) : Boolean
5233:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5234:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5235:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5236:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5237:  Function GetTextFromFile( Filename : String) : string
5238:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5239:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5240:  Function msStrToInt( s : String) : Integer
5241:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5242: end;
5243:
5244: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5245: begin
5246:   //TDynFloatArray', 'array of Extended
5247:   TDynLWordArray', 'array of LongWord
5248:   TDynLIntArray', 'array of LongInt
5249:   TDynFloatMatrix', 'array of TDynFloatArray
5250:   TDynLWordMatrix', 'array of TDynLWordArray
5251:   TDynLIntMatrix', 'array of TDynLIntArray
5252:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5253:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5254:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5255:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5256:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5257:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5258:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5259:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5260:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5261:  Function MNorm( const X : TDynFloatArray) : Extended
5262:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5263:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5264:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5265:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5266:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5267:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5268:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5269:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5270:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5271:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5272:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5273:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5274:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5275: end;
5276:
5277: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5278: begin
5279:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5280:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5281:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5282:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5283:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5284:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5285:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5286:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5287:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5288:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5289:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5290:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5291:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5292:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5293:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5294:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5295:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5296:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5297:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5298:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5299:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5300:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5301:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5302:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5303:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5304:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5305:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5306:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5307:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5308:  'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5309:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5310:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5311:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5312:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5313:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5314:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5315:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5316:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
```

```
5317:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5318:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5319:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5320:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5321:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5322:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5323:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5324:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5325:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5326:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5327:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5328:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5329:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5330:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5331:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5332:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5333:   //LongWord', 'Cardinal
5334:   TBitList', 'Word
5335:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5336:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5337:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5338:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5339:  Function SameFloat( const X1, X2 : Extended) : Boolean
5340:  Function FloatIsZero( const X : Extended) : Boolean
5341:  Function FloatIsPositive( const X : Extended) : Boolean
5342:  Function FloatIsNegative( const X : Extended) : Boolean
5343:  Procedure IncLim( var B : Byte; const Limit : Byte)
5344:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5345:  Procedure IncLimW( var B : Word; const Limit : Word)
5346:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5347:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5348:  Procedure DecLim( var B : Byte; const Limit : Byte)
5349:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5350:  Procedure DecLimW( var B : Word; const Limit : Word)
5351:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5352:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5353:  Function MaxB( const B1, B2 : Byte) : Byte
5354:  Function MinB( const B1, B2 : Byte) : Byte
5355:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5356:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5357:  Function MaxW( const B1, B2 : Word) : Word
5358:  Function MinW( const B1, B2 : Word) : Word
5359:  Function esbMaxI( const B1, B2 : Integer) : Integer
5360:  Function esbMinI( const B1, B2 : Integer) : Integer
5361:  Function MaxL( const B1, B2 : LongInt) : LongInt
5362:  Function MinL( const B1, B2 : LongInt) : LongInt
5363:  Procedure SwapB( var B1, B2 : Byte)
5364:  Procedure SwapSI( var B1, B2 : ShortInt)
5365:  Procedure SwapW( var B1, B2 : Word)
5366:  Procedure SwapI( var B1, B2 : SmallInt)
5367:  Procedure SwapL( var B1, B2 : LongInt)
5368:  Procedure SwapI32( var B1, B2 : Integer)
5369:  Procedure SwapC( var B1, B2 : LongWord)
5370:  Procedure SwapInt64( var X, Y : Int64)
5371:  Function esbSign( const B : LongInt) : ShortInt
5372:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5373:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5374:  Function Max3Word( const X1, X2, X3 : Word) : Word
5375:  Function Min3Word( const X1, X2, X3 : Word) : Word
5376:  Function MaxBArray( const B : array of Byte) : Byte
5377:  Function MaxWArray( const B : array of Word) : Word
5378:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5379:  Function MaxIArray( const B : array of Integer) : Integer
5380:  Function MaxLArray( const B : array of LongInt) : LongInt
5381:  Function MinBArray( const B : array of Byte) : Byte
5382:  Function MinWArray( const B : array of Word) : Word
5383:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5384:  Function MinIArray( const B : array of Integer) : Integer
5385:  Function MinLArray( const B : array of LongInt) : LongInt
5386:  Function SumBArray( const B : array of Byte) : Byte
5387:  Function SumBArray2( const B : array of Byte) : Word
5388:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5389:  Function SumSIArray2( const B : array of ShortInt) : Integer
5390:  Function SumWArray( const B : array of Word) : Word
5391:  Function SumWArray2( const B : array of Word) : LongInt
5392:  Function SumIArray( const B : array of Integer) : Integer
5393:  Function SumLArray( const B : array of LongInt) : LongInt
5394:  Function SumLWArray( const B : array of LongWord) : LongWord
5395:  Function ESBDigits( const X : LongWord) : Byte
5396:  Function BitsHighest( const X : LongWord) : Integer
5397:  Function ESBBitsNeeded( const X : LongWord) : Integer
5398:  Function esbGCD( const X, Y : LongWord) : LongWord
5399:  Function esbLCM( const X, Y : LongInt) : Int64
5400:  //Function esbLCM( const X, Y : LongInt) : LongInt
5401:  Function RelativePrime( const X, Y : LongWord) : Boolean
5402:  Function Get87ControlWord : TBitList
5403:  Procedure Set87ControlWord( const CWord : TBitList)
5404:  Procedure SwapExt( var X, Y : Extended)
5405:  Procedure SwapDbl( var X, Y : Double)
```

```
5406:  Procedure SwapSing( var X, Y : Single)
5407:  Function esbSgn( const X : Extended) : ShortInt
5408:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5409:  Function ExtMod( const X, Y : Extended) : Extended
5410:  Function ExtRem( const X, Y : Extended) : Extended
5411:  Function CompMOD( const X, Y : Comp) : Comp
5412:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5413:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5414:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5415:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5416:  Function MaxExt( const X, Y : Extended) : Extended
5417:  Function MinExt( const X, Y : Extended) : Extended
5418:  Function MaxEArray( const B : array of Extended) : Extended
5419:  Function MinEArray( const B : array of Extended) : Extended
5420:  Function MaxSArray( const B : array of Single) : Single
5421:  Function MinSArray( const B : array of Single) : Single
5422:  Function MaxCArray( const B : array of Comp) : Comp
5423:  Function MinCArray( const B : array of Comp) : Comp
5424:  Function SumSArray( const B : array of Single) : Single
5425:  Function SumEArray( const B : array of Extended) : Extended
5426:  Function SumSqEArray( const B : array of Extended) : Extended
5427:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5428:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5429:  Function SumCArray( const B : array of Comp) : Comp
5430:  Function FactorialX( A : LongWord) : Extended
5431:  Function PermutationX( N, R : LongWord) : Extended
5432:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5433:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5434:  Function esbGeometricMean( const X : array of Extended) : Extended
5435:  Function esbHarmonicMean( const X : array of Extended) : Extended
5436:  Function ESBMean( const X : array of Extended) : Extended
5437:  Function esbSampleVariance( const X : array of Extended) : Extended
5438:  Function esbPopulationVariance( const X : array of Extended) : Extended
5439:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5440:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5441:  Function GetMedian( const SortedX : array of Extended) : Extended
5442:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5443:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5444:  Function ESBMagnitude( const X : Extended) : Integer
5445:  Function ESBTan( Angle : Extended) : Extended
5446:  Function ESBCot( Angle : Extended) : Extended
5447:  Function ESBCosec( const Angle : Extended) : Extended
5448:  Function ESBSec( const Angle : Extended) : Extended
5449:  Function ESBArcTan( X, Y : Extended) : Extended
5450:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5451:  Function ESBArcCos( const X : Extended) : Extended
5452:  Function ESBArcSin( const X : Extended) : Extended
5453:  Function ESBArcSec( const X : Extended) : Extended
5454:  Function ESBArcCosec( const X : Extended) : Extended
5455:  Function ESBLog10( const X : Extended) : Extended
5456:  Function ESBLog2( const X : Extended) : Extended
5457:  Function ESBLogBase( const X, Base : Extended) : Extended
5458:  Function Pow2( const X : Extended) : Extended
5459:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5460:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5461:  Function XtoY( const X, Y : Extended) : Extended
5462:  Function esbTenToY( const Y : Extended) : Extended
5463:  Function esbTwoToY( const Y : Extended) : Extended
5464:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5465:  Function esbISqrt( const I : LongWord) : Longword
5466:  Function ILog2( const I : LongWord) : LongWord
5467:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5468:  Function ESBArCosh( X : Extended) : Extended
5469:  Function ESBArSinh( X : Extended) : Extended
5470:  Function ESBArTanh( X : Extended) : Extended
5471:  Function ESBCosh( X : Extended) : Extended
5472:  Function ESBSinh( X : Extended) : Extended
5473:  Function ESBTanh( X : Extended) : Extended
5474:  Function InverseGamma( const X : Extended) : Extended
5475:  Function esbGamma( const X : Extended) : Extended
5476:  Function esbLnGamma( const X : Extended) : Extended
5477:  Function esbBeta( const X, Y : Extended) : Extended
5478:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5479: end;
5480:
5481: ******************************Integer Huge Cardinal Utils*************************
5482: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5483: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5484: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5485: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5486: Function BitCount_8( Value : byte) : integer
5487: Function BitCount_16( Value : uint16) : integer
5488: Function BitCount_32( Value : uint32) : integer
5489: Function BitCount_64( Value : uint64) : integer
5490: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5491:  Procedure ( CountPrimalityTests : integer)
5492:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5493:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5494:  Function isCoPrime( a, b : THugeCardinal) : boolean
```

```
5495:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5496:  Function hasSmallFactor( p : THugeCardinal) : boolean
5497:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
       TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
       NumbersTested: integer) : boolean
5498:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5499:  Const('StandardExponent','LongInt'( 65537);
5500: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
       Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
       Numbers
5501:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5502:
5503: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5504: begin
5505:   AddTypeS('TXRTLInteger', 'array of Integer
5506:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5507:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5508:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5509:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5510:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5511:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5512:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5513:   'BitsPerByte','LongInt'( 8);
5514:   BitsPerDigit','LongInt'( 32);
5515:   SignBitMask','LongWord( $80000000);
5516:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5517:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5518:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5519:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5520:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5521:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5522:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5523:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5524:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5525:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5526:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5527:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5528:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5529:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5530:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5531:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5532:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5533:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5534:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5535:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5536:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5537:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5538:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5539:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5540:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5541:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5542:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5543:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5544:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5545:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5546:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5547:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5548:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5549:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5550:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5551:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5552:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5553:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
       AHighApproxResult:TXRTLInteger)
5554:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
       AHighApproxResult:TXRTLInteger);
5555:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5556:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5557:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5558:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5559:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5560:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5561:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5562:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5563:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5564:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5565:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5566:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5567:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5568:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5569:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5570:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5571:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5572:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5573:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5574:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5575:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5576:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5577:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
```

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

```
5660:  Function BinaryBS( B : Byte) : ShortString
5661:  Function BinaryWS( W : Word) : ShortString
5662:  Function BinaryLS( L : LongInt) : ShortString
5663:  Function OctalBS( B : Byte) : ShortString
5664:  Function OctalWS( W : Word) : ShortString
5665:  Function OctalLS( L : LongInt) : ShortString
5666:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5667:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5668:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5669:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5670:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5671:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5672:  Function Long2StrS( L : LongInt) : ShortString
5673:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5674:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5675:  Function ValPrepS( const S : ShortString) : ShortString
5676:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5677:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5678:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5679:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5680:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5681:  Function TrimLeadS( const S : ShortString) : ShortString
5682:  Function TrimTrailS( const S : ShortString) : ShortString
5683:  Function TrimS( const S : ShortString) : ShortString
5684:  Function TrimSpacesS( const S : ShortString) : ShortString
5685:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5686:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5687:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5688:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5689:  Function ScrambleS( const S, Key : ShortString) : ShortString
5690:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5691:  Function FilterS( const S, Filters : ShortString) : ShortString
5692:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5693:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5694:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5695:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5696:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5697:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5698:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5699:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5700:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5701:  Function CompStringS( const S1, S2 : ShortString) : Integer
5702:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5703:  Function SoundexS( const S : ShortString) : ShortString
5704:  Function MakeLetterSetS( const S : ShortString) : Longint
5705:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5706:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5707:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
      Pos:Cardinal):Bool;
5708:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5709:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5710:  Function JustFilenameS( const PathName : ShortString) : ShortString
5711:  Function JustNameS( const PathName : ShortString) : ShortString
5712:  Function JustExtensionS( const Name : ShortString) : ShortString
5713:  Function JustPathnameS( const PathName : ShortString) : ShortString
5714:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5715:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5716:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5717:  Function CommaizeS( L : LongInt) : ShortString
5718:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5719:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
      DecPt:Char):ShortString;
5720:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5721:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5722:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5723:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5724:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5725:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5726:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5727:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5728:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5729:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5730:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5731:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5732:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5733:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5734:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
5735:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
      SubStr:ShortString):Bool;
5736:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5737:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5738:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5739:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
```

```
5740:  Function ExtractTokensS(const S,
      Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5741:  Function IsChAlphaS( C : Char) : Boolean
5742:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5743:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5744:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5745:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5746:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5747:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5748:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5749:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5750:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5751:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5752:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
      Cardinal):ShortString;
5753:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
      Replacements:Cardinal):ShortString;
5754:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5755:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
      Replacements:Cardinal):ShortString
5756:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
      Replacements:Cardinal):ShortString
5757:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5758:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5759:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5760:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5761: end;
5762:
5763:
5764: ********unit uPSI_StUtils; from Systools4*****************************************************
5765: Function SignL( L : LongInt) : Integer
5766: Function SignF( F : Extended) : Integer
5767: Function MinWord( A, B : Word) : Word
5768: Function MidWord( W1, W2, W3 : Word) : Word
5769: Function MaxWord( A, B : Word) : Word
5770: Function MinLong( A, B : LongInt) : LongInt
5771: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5772: Function MaxLong( A, B : LongInt) : LongInt
5773: Function MinFloat( F1, F2 : Extended) : Extended
5774: Function MidFloat( F1, F2, F3 : Extended) : Extended
5775: Function MaxFloat( F1, F2 : Extended) : Extended
5776: Function MakeInteger16( H, L : Byte) : SmallInt
5777: Function MakeWordS( H, L : Byte) : Word
5778: Function SwapNibble( B : Byte) : Byte
5779: Function SwapWord( L : LongInt) : LongInt
5780: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5781: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5782: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5783: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5784: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5785: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5786: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5787: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5788: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5789: Procedure ExchangeBytes( var I, J : Byte)
5790: Procedure ExchangeWords( var I, J : Word)
5791: Procedure ExchangeLongInts( var I, J : LongInt)
5792: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5793: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5794: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5795: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5796: //*****************uPSI_StFIN;************************************************************
5797: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5798: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
      Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5799: Function BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5800: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
      Extended
5801: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5802: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5803: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5804: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5805: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5806: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5807: Function DollarToDecimalText( DecDollar : Extended) : string
5808: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5809: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5810: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5811: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5812: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5813: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5814: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5815: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5816: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
```

```
5817: Function IsCardValid( const S : string) : Boolean
5818: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5819: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5820: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5821: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5822: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5823: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5824: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5825: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5826: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5827: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5828: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5829: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5830: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5831: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5832: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5833: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5834: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5835: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5836: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5837: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5838:
5839: //*****************************************unit uPSI_StAstroP;
5840: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5841: //*****unit unit uPSI_StStat; Statistic Package of SysTools***************************************
5842: Function AveDev( const Data : array of Double) : Double
5843: Function AveDev16( const Data, NData : Integer) : Double
5844: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5845: Function Correlation( const Data1, Data2 : array of Double) : Double
5846: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5847: Function Covariance( const Data1, Data2 : array of Double) : Double
5848: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5849: Function DevSq( const Data : array of Double) : Double
5850: Function DevSq16( const Data, NData : Integer) : Double
5851: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5852:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5853: Function GeometricMeanS( const Data : array of Double) : Double
5854: Function GeometricMean16( const Data, NData : Integer) : Double
5855: Function HarmonicMeanS( const Data : array of Double) : Double
5856: Function HarmonicMean16( const Data, NData : Integer) : Double
5857: Function Largest( const Data : array of Double; K : Integer) : Double
5858: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5859: Function MedianS( const Data : array of Double) : Double
5860: Function Median16( const Data, NData : Integer) : Double
5861: Function Mode( const Data : array of Double) : Double
5862: Function Mode16( const Data, NData : Integer) : Double
5863: Function Percentile( const Data : array of Double; K : Double) : Double
5864: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5865: Function PercentRank( const Data : array of Double; X : Double) : Double
5866: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5867: Function Permutations( Number, NumberChosen : Integer) : Extended
5868: Function Combinations( Number, NumberChosen : Integer) : Extended
5869: Function FactorialS( N : Integer) : Extended
5870: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5871: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5872: Function Smallest( const Data : array of Double; K : Integer) : Double
5873: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5874: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5875: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5876:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5877:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5878: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5879: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5880: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5881: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5882: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5883: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5884: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5885: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5886: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5887: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5888: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5889: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5890: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5891: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5892: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5893: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5894: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5895: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5896: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5897: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5898: Function NormInv( Probability, Mean, StandardDev : Single) : Single
```

```
5899: Function NormSDist( Z : Single) : Single
5900: Function NormSInv( Probability : Single) : Single
5901: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5902: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5903: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5904: Function Erfc( X : Single) : Single
5905: Function GammaLn( X : Single) : Single
5906: Function LargestSort( const Data : array of Double; K : Integer) : Double
5907: Function SmallestSort( const Data : array of double; K : Integer) : Double
5908:
5909: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5910:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5911:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5912:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5913:  Function DefaultMergeName( MergeNum : Integer) : string
5914:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5915:
5916: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5917: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5918:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5919:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5920:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5921:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5922:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5923:  Function LunarPhase( UT : TStDateTimeRec) : Double
5924:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5925:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5926:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5927:  Function FullMoon( D : TStDate) : TStLunarRecord
5928:  Function LastQuarter( D : TStDate) : TStLunarRecord
5929:  Function NewMoon( D : TStDate) : TStLunarRecord
5930:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5931:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5932:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5933:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5934:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5935:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5936:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5937:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5938:  Function SiderealTime( UT : TStDateTimeRec) : Double
5939:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5940:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5941:  Function SEaster( Y, Epoch : Integer) : TStDate
5942:  Function DateTimeToAJD( D : TDateTime) : Double
5943:  Function HoursMin( RA : Double) : ShortString
5944:  Function DegsMin( DC : Double) : ShortString
5945:  Function AJDToDateTime( D : Double) : TDateTime
5946:
5947: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5948:  Function CurrentDate : TStDate
5949:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5950:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5951:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5952:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5953:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5954:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5955:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5956:  Function WeekOfYear( Julian : TStDate) : Byte
5957:  Function AstJulianDate( Julian : TStDate) : Double
5958:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5959:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5960:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5961:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5962:  Function StIsLeapYear( Year : Integer) : Boolean
5963:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5964:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5965:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5966:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5967:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5968:  Function CurrentTime : TStTime
5969:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5970:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5971:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5972:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5973:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5974:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5975:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5976:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5977:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5978:  Function StDateToDateTime( D : TStDate) : TDateTime
5979:  Function StTimeToDateTime( T : TStTime) : TDateTime
5980:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5981:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5982:
5983: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5984: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5985:  Function MonthToString( const Month : Integer) : string
5986:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5987:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
```

```
5988:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5989:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5990:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5991:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5992:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5993:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5994:  Function TimeStringToStTime( const Picture, S : string) : TStTime
5995:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5996:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5997:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5998:  Function InternationalDate( ForceCentury : Boolean) : string
5999:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
6000:  Function InternationalTime( ShowSeconds : Boolean) : string
6001:  Procedure ResetInternationalInfo
6002:
6003: procedure SIRegister_StBase(CL: TPSPascalCompiler);
6004:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
6005:  Function AnsiUpperCaseShort32( const S : string) : string
6006:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
6007:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
6008:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
6009:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
6010:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
6011:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
6012:  Function Upcase( C : AnsiChar) : AnsiChar
6013:  Function LoCase( C : AnsiChar) : AnsiChar
6014:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
6015:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
6016:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6017:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6018:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6019:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
6020:  Procedure RaiseContainerError( Code : longint)
6021:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6022:  Function ProductOverflow( A, B : LongInt) : Boolean
6023:  Function StNewStr( S : string) : PShortString
6024:  Procedure StDisposeStr( PS : PShortString)
6025:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
6026:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
6027:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6028:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6029:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6030:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6031:
6032: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6033: begin
6034:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6035:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6036:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6037:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6038:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6039: end;
6040:
6041: //**********unit unit ; StMath Package of SysTools*****************************************
6042: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6043: Function PowerS( Base, Exponent : Extended) : Extended
6044: Function StInvCos( X : Double) : Double
6045: Function StInvSin( Y : Double) : Double
6046: Function StInvTan2( X, Y : Double) : Double
6047: Function StTan( A : Double) : Double
6048: Procedure DumpException;  //unit StExpEng;
6049: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6050:
6051: //**********unit unit ; StCRC Package of SysTools*****************************************
6052: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6053: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6054: Function Adler32OfFile( FileName : AnsiString) : LongInt
6055: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6056: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6057: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6058: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6059: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6060: Function Crc32OfFile( FileName : AnsiString) : LongInt
6061: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6062: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6063: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6064: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6065: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6066: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6067:
6068: //**********unit unit ; StBCD Package of SysTools*****************************************
6069: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6070: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6071: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6072: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6073: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6074: Function NegBcd( const B : TbcdS) : TbcdS
6075: Function AbsBcd( const B : TbcdS) : TbcdS
6076: Function FracBcd( const B : TbcdS) : TbcdS
```

```
6077: Function IntBcd( const B : TbcdS) : TbcdS
6078: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6079: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6080: Function ValBcd( const S : string) : TbcdS
6081: Function LongBcd( L : LongInt) : TbcdS
6082: Function ExtBcd( E : Extended) : TbcdS
6083: Function ExpBcd( const B : TbcdS) : TbcdS
6084: Function LnBcd( const B : TbcdS) : TbcdS
6085: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6086: Function PowBcd( const B, E : TbcdS) : TbcdS
6087: Function SqrtBcd( const B : TbcdS) : TbcdS
6088: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6089: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6090: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6091: Function IsIntBcd( const B : TbcdS) : Boolean
6092: Function TruncBcd( const B : TbcdS) : LongInt
6093: Function BcdExt( const B : TbcdS) : Extended
6094: Function RoundBcd( const B : TbcdS) : LongInt
6095: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6096: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6097: Function FormatBcd( const Format : string; const B : TbcdS) : string
6098: Function StrGeneralBcd( const B : TbcdS) : string
6099: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6100: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6101:
6102: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6103: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6104: Function StFieldTypeToStr( FieldType : TStStSchemaFieldType) : AnsiString
6105: Function StStrToFieldType( const S : AnsiString) : TStStSchemaFieldType
6106: Function StDeEscape( const EscStr : AnsiString) : Char
6107: Function StDoEscape( Delim : Char) : AnsiString
6108: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6109: Function AnsiHashText( const S : string; Size : Integer) : Integer
6110: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6111: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6112: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6113:
6114: //**********unit unit ; StNetCon Package of SysTools*****************************************
6115:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6116:     Constructor Create( AOwner : TComponent)
6117:     Function Connect : DWord
6118:     Function Disconnect : DWord
6119:     RegisterProperty('Password', 'String', iptrw);
6120:     Property('UserName', 'String', iptrw);
6121:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6122:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6123:     Property('LocalDevice', 'String', iptrw);
6124:     Property('ServerName', 'String', iptrw);
6125:     Property('ShareName', 'String', iptrw);
6126:     Property('OnConnect', 'TNotifyEvent', iptrw);
6127:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6128:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6129:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6130:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6131:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6132:   end;
6133: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6134: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6135: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6136: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6137: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6138: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6139: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6140: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6141: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6142: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6143: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6144: Function SuspendThread( hThread : THandle) : DWORD
6145: Function ResumeThread( hThread : THandle) : DWORD
6146: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6147: Function GetCurrentThread : THandle
6148: Procedure ExitThread( dwExitCode : DWORD)
6149: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6150: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6151: Procedure EndThread(ExitCode: Integer);
6152: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6153: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6154: Procedure FreeProcInstance( Proc : FARPROC)
6155: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6156: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6157: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6158: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6159: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6160: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6161: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6162: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6163: Function CurrentParallelJobInfo : TParallelJobInfo
6164: Function ObtainParallelJobInfo : TParallelJobInfo
```

```
6165: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6166: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6167: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6168: Function
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL');
6169: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL');
6170: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD) : BOOL');
6171: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6172: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6173:
6174: ************************************************unit uPSI_JclMime;
6175:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6176:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6177:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6178:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6179:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6180:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6181:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6182:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6183:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6184:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6185:
6186: ************************************************unit uPSI_JclPrint;
6187:  Procedure DirectPrint( const Printer, Data : string)
6188:  Procedure SetPrinterPixelsPerInch
6189:  Function GetPrinterResolution : TPoint
6190:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6191:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6192:
6193:
6194: //*************************************unit uPSI_ShLwApi;***********************************
6195:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6196:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6197:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6198:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6199:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6200:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6201:  Function StrDup( lpSrch : PChar) : PChar
6202:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6203:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6204:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6205:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6206: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6207:  Function StrPBrk( psz, pszSet : PChar) : PChar
6208:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6209:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6210:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6211:  Function StrSpn( psz, pszSet : PChar) : Integer
6212:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6213:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6214:  Function StrToInt( lpSrch : PChar) : Integer
6215:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6216:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6217:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6218:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6219:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6220:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6221:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6222:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6223:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6224:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6225:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6226:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6227:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6228:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6229:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6230:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6231:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6232:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6233:  STIF_DEFAULT','LongWord( $00000000);
6234:  STIF_SUPPORT_HEX','LongWord( $00000001);
6235:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6236:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6237:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6238:  Function PathAddBackslash( pszPath : PChar) : PChar
6239:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6240:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6241:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6242:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6243:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6244:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6245:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6246:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6247:  Function PathFileExists( pszPath : PChar) : BOOL
6248:  Function PathFindExtension( pszPath : PChar) : PChar
```

```
6249:  Function PathFindFileName( pszPath : PChar) : PChar
6250:  Function PathFindNextComponent( pszPath : PChar) : PChar
6251:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6252:  Function PathGetArgs( pszPath : PChar) : PChar
6253:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6254:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6255:  Function PathGetCharType( ch : Char) : UINT
6256:  GCT_INVALID','LongWord( $0000);
6257:  GCT_LFNCHAR','LongWord( $0001);
6258:  GCT_SHORTCHAR','LongWord( $0002);
6259:  GCT_WILD','LongWord( $0004);
6260:  GCT_SEPARATOR','LongWord( $0008);
6261:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6262:  Function PathIsDirectory( pszPath : PChar) : BOOL
6263:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6264:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6265:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6266:  Function PathIsRelative( pszPath : PChar) : BOOL
6267:  Function PathIsRoot( pszPath : PChar) : BOOL
6268:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6269:  Function PathIsUNC( pszPath : PChar) : BOOL
6270:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6271:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6272:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6273:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6274:  Function PathIsURL( pszPath : PChar) : BOOL
6275:  Function PathMakePretty( pszPath : PChar) : BOOL
6276:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6277:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6278:  Procedure PathQuoteSpaces( lpsz : PChar)
6279:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6280:  Procedure PathRemoveArgs( pszPath : PChar)
6281:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6282:  Procedure PathRemoveBlanks( pszPath : PChar)
6283:  Procedure PathRemoveExtension( pszPath : PChar)
6284:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6285:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6286:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6287:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6288:  Function PathSkipRoot( pszPath : PChar) : PChar
6289:  Procedure PathStripPath( pszPath : PChar)
6290:  Function PathStripToRoot( pszPath : PChar) : BOOL
6291:  Procedure PathUnquoteSpaces( lpsz : PChar)
6292:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6293:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6294:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6295:  Procedure PathUndecorate( pszPath : PChar)
6296:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6297:  URL_SCHEME_INVALID','LongInt'( - 1);
6298:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6299:  URL_SCHEME_FTP','LongInt'( 1);
6300:  URL_SCHEME_HTTP','LongInt'( 2);
6301:  URL_SCHEME_GOPHER','LongInt'( 3);
6302:  URL_SCHEME_MAILTO','LongInt'( 4);
6303:  URL_SCHEME_NEWS','LongInt'( 5);
6304:  URL_SCHEME_NNTP','LongInt'( 6);
6305:  URL_SCHEME_TELNET','LongInt'( 7);
6306:  URL_SCHEME_WAIS','LongInt'( 8);
6307:  URL_SCHEME_FILE','LongInt'( 9);
6308:  URL_SCHEME_MK','LongInt'( 10);
6309:  URL_SCHEME_HTTPS','LongInt'( 11);
6310:  URL_SCHEME_SHELL','LongInt'( 12);
6311:  URL_SCHEME_SNEWS','LongInt'( 13);
6312:  URL_SCHEME_LOCAL','LongInt'( 14);
6313:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6314:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6315:  URL_SCHEME_ABOUT','LongInt'( 17);
6316:  URL_SCHEME_RES','LongInt'( 18);
6317:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6318:  URL_SCHEME', 'Integer
6319:  URL_PART_NONE','LongInt'( 0);
6320:  URL_PART_SCHEME','LongInt'( 1);
6321:  URL_PART_HOSTNAME','LongInt'( 2);
6322:  URL_PART_USERNAME','LongInt'( 3);
6323:  URL_PART_PASSWORD','LongInt'( 4);
6324:  URL_PART_PORT','LongInt'( 5);
6325:  URL_PART_QUERY','LongInt'( 6);
6326:  URL_PART', 'DWORD
6327:  URLIS_URL','LongInt'( 0);
6328:  URLIS_OPAQUE','LongInt'( 1);
6329:  URLIS_NOHISTORY','LongInt'( 2);
6330:  URLIS_FILEURL','LongInt'( 3);
6331:  URLIS_APPLIABLE','LongInt'( 4);
6332:  URLIS_DIRECTORY','LongInt'( 5);
6333:  URLIS_HASQUERY','LongInt'( 6);
6334:  TUrlIs', 'DWORD
6335:  URL_UNESCAPE','LongWord( $10000000);
6336:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6337:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
```

```
6338:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6339:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6340:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6341:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6342:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6343:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6344:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6345:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6346:  URL_INTERNAL_PATH','LongWord( $00800000);
6347:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6348:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6349:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6350:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6351:  URL_APPLY_DEFAULT','LongWord( $00000001);
6352:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6353:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6354:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6355:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6356:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6357:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6358:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6359:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6360:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6361:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6362:  Function UrlGetLocation( psz1 : PChar) : PChar
6363:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6364:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6365:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6366:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6367:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6368:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6369:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6370:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6371:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6372:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6373:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6374:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6375:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6376:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6377:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6378:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6379:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6380:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6381:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6382:  SHREGDEL_DEFAULT','LongWord( $00000000);
6383:  SHREGDEL_HKCU','LongWord( $00000001);
6384:  SHREGDEL_HKLM','LongWord( $00000010);
6385:  SHREGDEL_BOTH','LongWord( $00000011);
6386:  SHREGENUM_DEFAULT','LongWord( $00000000);
6387:  SHREGENUM_HKCU','LongWord( $00000001);
6388:  SHREGENUM_HKLM','LongWord( $00000010);
6389:  SHREGENUM_BOTH','LongWord( $00000011);
6390:  SHREGSET_HKCU','LongWord( $00000001);
6391:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6392:  SHREGSET_HKLM','LongWord( $00000004);
6393:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6394:  TSHRegDelFlags', 'DWORD
6395:  TSHRegEnumFlags', 'DWORD
6396:  HUSKEY', 'THandle
6397:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6398:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6399:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6400:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6401:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6402:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6403:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6404:  ASSOCF_VERIFY','LongWord( $00000040);
6405:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6406:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6407:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6408:  ASSOCF', 'DWORD
6409:  ASSOCSTR_COMMAND','LongInt'( 1);
6410:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6411:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6412:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6413:  ASSOCSTR_NOOPEN','LongInt'( 5);
6414:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6415:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6416:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6417:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6418:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6419:  ASSOCSTR_INFOTIP','LongInt'( 11);
6420:  ASSOCSTR_MAX','LongInt'( 12);
6421:  ASSOCSTR', 'DWORD
6422:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6423:  ASSOCKEY_APP','LongInt'( 2);
6424:  ASSOCKEY_CLASS','LongInt'( 3);
6425:  ASSOCKEY_BASECLASS','LongInt'( 4);
```

```
6426:  ASSOCKEY_MAX','LongInt'( 5);
6427:  ASSOCKEY', 'DWORD
6428:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6429:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6430:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6431:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6432:  ASSOCDATA_MAX','LongInt'( 5);
6433:  ASSOCDATA', 'DWORD
6434:  ASSOCENUM_NONE','LongInt'( 0);
6435:  ASSOCENUM', 'DWORD
6436:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6437:  SHACF_DEFAULT $00000000);
6438:  SHACF_FILESYSTEM','LongWord( $00000001);
6439:  SHACF_URLHISTORY','LongWord( $00000002);
6440:  SHACF_URLMRU','LongWord( $00000004);
6441:  SHACF_USETAB','LongWord( $00000008);
6442:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6443:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6444:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6445:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6446:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6447:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6448:  Procedure SHSetThreadRef( punk : IUnknown)
6449:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6450:  CTF_INSIST','LongWord( $00000001);
6451:  CTF_THREAD_REF','LongWord( $00000002);
6452:  CTF_PROCESS_REF','LongWord( $00000004);
6453:  CTF_COINIT','LongWord( $00000008);
6454:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6455:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6456:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6457:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6458:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6459:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6460:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6461:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6462:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6463:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6464:  Function SetRectEmpty( var lprc : TRect) : BOOL
6465:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6466:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6467:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6468:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6469:
6470:  Function InitializeFlatSB( hWnd : HWND) : Bool
6471:  Procedure UninitializeFlatSB( hWnd : HWND)
6472:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6473:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6474:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6475:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6476:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6477:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6478:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6479:
6480:
6481: //  ****************************************** 204 unit uPSI_ShellAPI;
6482: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6483: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6484: Procedure DragFinish( Drop : HDROP)
6485: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6486: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6487: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6488: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6489: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6490: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6491: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6492: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6493: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6494: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6495: Procedure SHFreeNameMappings( hNameMappings : THandle)
6496:
6497:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6498:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6499:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6500:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6501:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6502:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6503:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6504:  Function SimpleXMLEncode( const S : string) : string
6505:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6506:  Function XMLEncode( const S : string) : string
6507:  Function XMLDecode( const S : string) : string
6508:  Function EntityEncode( const S : string) : string
6509:  Function EntityDecode( const S : string) : string
6510:
6511: procedure RIRegister_CPort_Routines(S: TPSExec);
6512:  Procedure EnumComPorts( Ports : TStrings)
6513:  Procedure ListComPorts( Ports : TStrings)
6514:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
```

```
6515:  Function GetComPorts: TStringlist;
6516:  Function StrToBaudRate( Str : string) : TBaudRate
6517:  Function StrToStopBits( Str : string) : TStopBits
6518:  Function StrToDataBits( Str : string) : TDataBits
6519:  Function StrToParity( Str : string) : TParityBits
6520:  Function StrToFlowControl( Str : string) : TFlowControl
6521:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6522:  Function StopBitsToStr( StopBits : TStopBits) : string
6523:  Function DataBitsToStr( DataBits : TDataBits) : string
6524:  Function ParityToStr( Parity : TParityBits) : string
6525:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6526:  Function ComErrorsToStr( Errors : TComErrors) : String
6527:
6528:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6529:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6530:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6531:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6532:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6533:  Function GetMessagePos : DWORD
6534:  Function GetMessageTime : Longint
6535:  Function GetMessageExtraInfo : Longint
6536:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6537:  Procedure JAddToRecentDocs( const Filename : string)
6538:  Procedure ClearRecentDocs
6539:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6540:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6541:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6542:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6543:  Function RecycleFile( FileToRecycle : string) : Boolean
6544:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6545:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6546:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6547:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6548:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6549:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6550:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6551:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6552:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6553:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6554:
6555: ****************************************** unit uPSI_JclPeImage;
6556:
6557:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6558: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6559:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6560:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6561:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6562:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6563:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6564:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6565:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6566:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Bool
6567:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6568:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6569:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6570:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
      LibraryName:string; IncludeLibNames : Boolean): Boolean
6571:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6572:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6573:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6574:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
      NamesList:TStrings):Bool
6575:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6576:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
      Descript:Bool):Bool;
6577:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6578:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6579:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6580:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6581:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6582:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6583:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
      PImageSectionHeader
```

```
6584:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6585:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6586:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
       ___Pointer;
6587:    SIRegister_TJclPeSectionStream(CL);
6588:    SIRegister_TJclPeMapImgHookItem(CL);
6589:    SIRegister_TJclPeMapImgHooks(CL);
6590:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6591:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6592:    Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6593:    TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6594:    TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6595:    TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6596:    TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6597:    TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6598:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6599:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6600:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6601:  Function PeBorUnmangleName3( const Name : string) : string;
6602:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6603:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6604:
6605:
6606: //***************** SysTools uPSI_StSystem; ************************************
6607:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6608:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6609:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6610:  //Procedure EnumerateDirectories(const
       StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6611:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6612:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6613:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6614:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6615:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6616:  Function FlushOsBuffers( Handle : Integer) : Boolean
6617:  Function GetCurrentUser : AnsiString
6618:  Function GetDiskClass( Drive : Char) : DiskClass
6619:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6620:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6621:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6622:    { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6623:  Function getDiskSpace2(const path: String; index: integer): int64;
6624:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6625:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6626:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6627:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6628:  Function GetLongPath( const APath : AnsiString) : AnsiString
6629:  Function GetMachineName : AnsiString
6630:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6631:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6632:  Function GetShortPath( const APath : AnsiString) : AnsiString
6633:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6634:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6635:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6636:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6637:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6638:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6639:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6640:  Function IsDriveReady( Drive : Char) : Boolean
6641:  Function IsFile( const FileName : AnsiString) : Boolean
6642:  Function IsFileArchive( const S : AnsiString) : Integer
6643:  Function IsFileHidden( const S : AnsiString) : Integer
6644:  Function IsFileReadOnly( const S : AnsiString) : Integer
6645:  Function IsFileSystem( const S : AnsiString) : Integer
6646:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6647:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6648:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6649:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6650:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6651:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6652:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6653:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6654:  Function ValidDrive( Drive : Char) : Boolean
6655:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6656:
6657: //****************************unit uPSI_JclLANMan;*****************************************
6658: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6659:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6660:  Function DeleteAccount( const Servername, Username : string) : Boolean
6661:  Function DeleteLocalAccount( Username : string) : Boolean
```

```
6662:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6663:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6664:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6665:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6666:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6667:  Function LocalGroupExists( const Group : string) : Boolean
6668:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6669:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6670:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6671:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6672:  Function IsLocalAccount( const AccountName : string) : Boolean
6673:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6674:  Function GetRandomString( NumChar : cardinal) : string
6675:
6676: //****************************unit uPSI_cUtils;*****************************************
6677: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6678: Function cIsWinNT : boolean
6679:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6680:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6681:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6682:  Function cGetShortName( FileName : string) : string
6683:  Procedure cShowError( Msg : String)
6684:  Function cCommaStrToStr( s : string; formatstr : string) : string
6685:  Function cIncludeQuoteIfSpaces( s : string) : string
6686:  Function cIncludeQuoteIfNeeded( s : string) : string
6687:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6688:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6689:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6690:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6691:  Function cCodeInstoStr( s : string) : string
6692:  Function cStrtoCodeIns( s : string) : string
6693:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6694:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6695:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6696:  Function cPointtoStr( const pt : TPoint) : string
6697:  Function cListtoStr( const List : TStrings) : string
6698:  Function ListtoStr( const List : TStrings) : string
6699:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6700:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6701:  Function cGetFileTyp( const FileName : string) : TUnitType
6702:  Function cGetExTyp( const FileName : string) : TExUnitType
6703:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6704:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6705:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6706:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6707:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6708:  Function cGenMakePath( FileName : String) : String;
6709:  Function cGenMakePath2( FileName : String) : String
6710:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6711:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6712:  Function cCalcMod( Count : Integer) : Integer
6713:  Function cGetVersionString( FileName : string) : string
6714:  Function cCheckChangeDir( var Dir : string) : boolean
6715:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6716:  Function cIsNumeric( s : string) : boolean
6717:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6718:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6719:  Function GetFileTyp( const FileName : string) : TUnitType
6720:  Function Atoi(const aStr: string): integer
6721:  Function Itoa(const aint: integer): string
6722:  Function Atof(const aStr: string): double');
6723:  Function Atol(const aStr: string): longint');
6724:
6725:
6726: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6727: begin
6728:   FindClass('TOBJECT'),'EHTTP
6729:   FindClass('TOBJECT'),'EHTTPParser
6730:   //AnsiCharSet', 'set of AnsiChar
6731:   AnsiStringArray', 'array of AnsiString
6732:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6733:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6734:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6735:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6736:    +'CustomMinVersion : Integer; end
6737:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6738:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6739:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6740:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6741:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6742:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6743:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6744:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6745:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6746:    +'nection, hntOrigin, hntKeepAlive )
6747:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6748:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
```

```
6749:    +' AnsiString; end
6750:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6751:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6752:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6753:   //PHTTPContentLength', '^THTTPContentLength // will not work
6754:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6755:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6756:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6757:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6758:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6759:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6760:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6761:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6762:    +' CustomStr : AnsiString; end
6763:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6764:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6765:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6766:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6767:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6768:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6769:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6770:    +'m; Custom : AnsiString; end
6771:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6772:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6773:    +' Custom : AnsiString; end
6774:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6775:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6776:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6777:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6778:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6779:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6780:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6781:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6782:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6783:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6784:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6785:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6786:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6787:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6788:    +'FieldEnum; List : array of THTTPContentEncoding; end
6789:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6790:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6791:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6792:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6793:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6794:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6795:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6796:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6797:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6798:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6799:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6800:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6801:    +'CustomFieldArray; Custom : AnsiString; end
6802:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6803:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6804:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6805:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6806:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6807:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6808:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6809:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6810:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6811:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6812:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6813:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6814:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6815:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6816:   THTTPFixedHeaders','array[0..42] of AnsiString
6817:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6818:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6819:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6820:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6821:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6822:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6823:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6824:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6825:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6826:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6827:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6828:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6829:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6830:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6831:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6832:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6833:    +' THTTPDateField; Age : THTTPAgeField; end
6834:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6835:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6836:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6837:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
```

```
6838:  Procedure InitHTTPRequest( var A : THTTPRequest)
6839:  Procedure InitHTTPResponse( var A : THTTPResponse)
6840:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6841:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6842:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6843:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6844:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6845:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6846:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6847:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6848:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6849:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6850:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6851:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6852:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6853:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6854:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6855:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6856:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6857:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6858:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6859:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6860:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6861:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6862:   THTTPStringOption', '( hsoNone )
6863:   THTTPStringOptions', 'set of THTTPStringOption
6864:   FindClass('TOBJECT'),'TAnsiStringBuilder
6865:
6866:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6867:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6868:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6869:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6870:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6871:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6872:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6873:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6874:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6875:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6876:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6877:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6878:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6879:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6880:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6881:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6882:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6883:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6884:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6885:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6886:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6887:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6888:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6889:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6890:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6891:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6892:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6893:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6894:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
      THTTPStrOptions);
6895:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6896:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
      P:THTTPStringOptions);
```

```
6897:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6898:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6899:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6900:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6901:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6902:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6903:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
      Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6904:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6905:   SIRegister_THTTPParser(CL);
6906:   FindClass('TOBJECT'),'THTTPContentDecoder
6907:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6908:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6909:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6910:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6911:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6912:   SIRegister_THTTPContentDecoder(CL);
6913:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6914:   FindClass('TOBJECT'),'THTTPContentReader
6915:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6916:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
      LogLevel:Int;
6917:   SIRegister_THTTPContentReader(CL);
6918:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6919:   FindClass('TOBJECT'),'THTTPContentWriter
6920:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6921:   SIRegister_THTTPContentWriter(CL);
6922:  Procedure SelfTestcHTTPUtils
6923: end;
6924:
6925: (*----------------------------------------------------------------------------*)
6926: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6927: begin
6928:  'TLSLibraryVersion','String '1.00
6929:  'TLSError_None','LongInt'( 0);
6930:  'TLSError_InvalidBuffer','LongInt'( 1);
6931:  'TLSError_InvalidParameter','LongInt'( 2);
6932:  'TLSError_InvalidCertificate','LongInt'( 3);
6933:  'TLSError_InvalidState','LongInt'( 4);
6934:  'TLSError_DecodeError','LongInt'( 5);
6935:  'TLSError_BadProtocol','LongInt'( 6);
6936:  Function TLSErrorMessage( const TLSError : Integer) : String
6937:   SIRegister_ETLSError(CL);
6938:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6939:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6940:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6941:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6942:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6943:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6944:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6945:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6946:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6947:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6948:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6949:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6950:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6951:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6952:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6953:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6954:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6955:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6956:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6957:   PTLSRandom', '^TTLSRandom // will not work
6958:  Procedure InitTLSRandom( var Random : TTLSRandom)
6959:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6960:  'TLSSessionIDMaxLen','LongInt'( 32);
6961:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6962:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6963:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6964:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6965:    +'; Signature : TTLSSignatureAlgorithm; end
6966:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6967:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6968:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6969:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6970:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6971:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6972:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6973:    +'nteger; Supported : Boolean; end
6974:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6975:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6976:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6977:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6978:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6979:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6980:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6981:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6982:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6983:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
```

```
6984:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
       Size:Int):AString;
6985:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
       Size:Integer):AnsiString
6986:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
       Size:Int):AnsiString;
6987:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
       Size:Int):AnsiString;
6988:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
       ClientRandom : AnsiString; const Size : Integer) : AnsiString
6989:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6990:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6991:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6992:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
       ServerRandom:AnsiString) : AnsiString
6993:    TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6994:     +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6995:     +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6996:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
       IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6997:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
       ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6998:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
6999:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7000:  Procedure SelfTestcTLSUtils
7001: end;
7002:
7003: (*----------------------------------------------------------------------------*)
7004: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
7005: begin
7006:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
       integer; disks : integer; mx : integer; my : integer; end
7007:  // pBoard', '^tBoard // will not work
7008:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
7009:  Function rCheckMove( color : byte; cx, cy : integer) : integer
7010:  //Function rDoStep( data : pBoard) : word
7011:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
7012: end;
7013:
7014: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7015: begin
7016: Function InEditMode( ADataset : TDataset) : Boolean
7017: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
7018: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7019: Function GetFieldText( AField : TField) : String
7020: end;
7021:
7022: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
7023: begin
7024:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7025:   TMyPrintRange', '( prAll, prSelected )
7026:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7027:    +'ded, ssDateTime, ssTime, ssCustom )
7028:   TSortDirection', '( sdAscending, sdDescending )
7029:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7030:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7031:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7032:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7033:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7034:   SIRegister_TSortOptions(CL);
7035:   SIRegister_TPrintOptions(CL);
7036:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7037:   SIRegister_TSortedList(CL);
7038:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7039:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7040:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7041:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7042:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7043:   SIRegister_TFontSetting(CL);
7044:   SIRegister_TFontList(CL);
7045:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7046:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7047:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7048:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7049:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7050:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7051:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7052:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7053:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
7054:    +'r; var SortStyle : TSortStyle)
7055:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
7056:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7057:   SIRegister_TSortGrid(CL);
7058:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7059:  Function NormalCompare( const Str1, Str2 : String) : Integer
7060:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7061:  Function NumericCompare( const Str1, Str2 : String) : Integer
7062:  Function TimeCompare( const Str1, Str2 : String) : Integer
7063:  //Function Compare( Item1, Item2 : Pointer) : Integer
```

```
7064: end;
7065:
7066: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7067:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7068:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7069:  Procedure IBDataBaseError
7070:  Function StatusVector : PISC_STATUS
7071:  Function StatusVectorArray : PStatusVector
7072:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7073:  Function StatusVectorAsText : string
7074:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7075:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7076:
7077:
7078: //*****************************unit uPSI_BoldUtils;*****************************************
7079:  Function CharCount( c : char; const s : string) : integer
7080:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7081:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7082:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7083:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7084:  Function BoldTrim( const S : string) : string
7085:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7086:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7087:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7088:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7089:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7090:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7091:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7092:  Function CapitalisedToSpaced( Capitalised : String) : String
7093:  Function SpacedToCapitalised( Spaced : String) : String
7094:  Function BooleanToString( BoolValue : Boolean) : String
7095:  Function StringToBoolean( StrValue : String) : Boolean
7096:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7097:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7098:  Function StringListToVarArray( List : TStringList) : variant
7099:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7100:  Function GetComputerNameStr : string
7101:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7102:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7103:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7104:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7105:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7106:  Procedure EnsureTrailing( var Str : String; ch : char)
7107:  Function BoldDirectoryExists( const Name : string) : Boolean
7108:  Function BoldForceDirectories( Dir : string) : Boolean
7109:  Function BoldRootRegistryKey : string
7110:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7111:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7112:  Function LogicalAnd( A, B : Integer) : Boolean
7113:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7114:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7115:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7116:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7117:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7118:  Function IsFirstInstance : Boolean
7119:  Procedure ActivateFirst( AString : PChar)
7120:  Procedure ActivateFirstCommandLine
7121:  function MakeAckPkt(const BlockNumber: Word): string;
7122:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7123:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7124:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7125:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7126:  function IdStrToWord(const Value: String): Word;
7127:  function IdWordToStr(const Value: Word): WordStr;
7128:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7129:  Function CPUFeatures : TCPUFeatures
7130:
7131: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7132: begin
7133:   AddTypeS('TXRTLBitIndex', 'Integer
7134:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7135:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7136:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7137:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7138:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7139:  Function XRTLSwapHiLo16( X : Word) : Word
7140:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7141:  Function XRTLSwapHiLo64( X : Int64) : Int64
7142:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7143:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7144:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7145:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7146:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7147:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7148:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7149:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7150:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7151:  Function XRTLPopulation( A : Cardinal) : Cardinal
```

```
7152: end;
7153:
7154: Function XRTLURLDecode( const ASrc : WideString) : WideString
7155: Function XRTLURLEncode( const ASrc : WideString) : WideString
7156: Function XRTLURINormalize( const AURI : WideString) : WideString
7157: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7158: Function XRTLExtractLongPathName(APath: string): string;
7159:
7160: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7161: begin
7162:   AddTypeS('Int8', 'ShortInt
7163:   AddTypeS('Int16', 'SmallInt
7164:   AddTypeS('Int32', 'LongInt
7165:   AddTypeS('UInt8', 'Byte
7166:   AddTypeS('UInt16', 'Word
7167:   AddTypeS('UInt32', 'LongWord
7168:   AddTypeS('UInt64', 'Int64
7169:   AddTypeS('Word8', 'UInt8
7170:   AddTypeS('Word16', 'UInt16
7171:   AddTypeS('Word32', 'UInt32
7172:   AddTypeS('Word64', 'UInt64
7173:   AddTypeS('LargeInt', 'Int64
7174:   AddTypeS('NativeInt', 'Integer
7175:   AddTypeS('NativeUInt', 'Cardinal
7176:   Const('BitsPerByte','LongInt'( 8);
7177:   Const('BitsPerWord','LongInt'( 16);
7178:   Const('BitsPerLongWord','LongInt'( 32);
7179:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7180:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7181:  Function MinI( const A, B : Integer) : Integer
7182:  Function MaxI( const A, B : Integer) : Integer
7183:  Function MinC( const A, B : Cardinal) : Cardinal
7184:  Function MaxC( const A, B : Cardinal) : Cardinal
7185:  Function SumClipI( const A, I : Integer) : Integer
7186:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7187:  Function InByteRange( const A : Int64) : Boolean
7188:  Function InWordRange( const A : Int64) : Boolean
7189:  Function InLongWordRange( const A : Int64) : Boolean
7190:  Function InShortIntRange( const A : Int64) : Boolean
7191:  Function InSmallIntRange( const A : Int64) : Boolean
7192:  Function InLongIntRange( const A : Int64) : Boolean
7193:   AddTypeS('Bool8', 'ByteBool
7194:   AddTypeS('Bool16', 'WordBool
7195:   AddTypeS('Bool32', 'LongBool
7196:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7197:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7198:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7199:  Const('MinSingle','Single').setExtended( 1.5E-45);
7200:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7201:  Const('MinDouble','Double').setExtended( 5.0E-324);
7202:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7203:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7204:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7205:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7206:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7207:  Function MinF( const A, B : Float) : Float
7208:  Function MaxF( const A, B : Float) : Float
7209:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7210:  Function InSingleRange( const A : Float) : Boolean
7211:  Function InDoubleRange( const A : Float) : Boolean
7212:  Function InCurrencyRange( const A : Float) : Boolean;
7213:  Function InCurrencyRange1( const A : Int64) : Boolean;
7214:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7215:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7216:  Function FloatIsInfinity( const A : Extended) : Boolean
7217:  Function FloatIsNaN( const A : Extended) : Boolean
7218:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7219:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7220:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7221:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7222:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7223:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7224:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7225:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7226:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7227:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7228:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7229:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7230:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7231:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7232:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7233:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7234:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7235:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7236:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7237:  Function SetBitScanForward( const Value : LongWord) : Integer;
7238:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7239:  Function SetBitScanReverse( const Value : LongWord) : Integer;
```

```
7240:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7241:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7242:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7243:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7244:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7245:  Function cReverseBits( const Value : LongWord) : LongWord;
7246:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7247:  Function cSwapEndian( const Value : LongWord) : LongWord
7248:  Function cTwosComplement( const Value : LongWord) : LongWord
7249:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7250:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7251:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7252:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7253:  Function cBitCount( const Value : LongWord) : LongWord
7254:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7255:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7256:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7257:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7258:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7259:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7260:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7261:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7262:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7263: //  AddTypeS('CharSet', 'set of AnsiChar
7264:   AddTypeS('CharSet', 'set of Char       //!!!
7265:   AddTypeS('AnsiCharSet', 'TCharSet
7266:   AddTypeS('ByteSet', 'set of Byte
7267:   AddTypeS('AnsiChar', 'Char
7268:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7269:  Function AsByteSet( const C : array of Byte) : ByteSet
7270:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7271:  Procedure ClearCharSet( var C : CharSet)
7272:  Procedure FillCharSet( var C : CharSet)
7273:  procedure FillCharSearchRec;  // with 0
7274:  Procedure ComplementCharSet( var C : CharSet)
7275:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7276:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7277:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7278:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7279:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7280:  Function IsSubSet( const A, B : CharSet) : Boolean
7281:  Function IsEqual( const A, B : CharSet) : Boolean
7282:  Function IsEmpty( const C : CharSet) : Boolean
7283:  Function IsComplete( const C : CharSet) : Boolean
7284:  Function cCharCount( const C : CharSet) : Integer
7285:  Procedure ConvertCaseInsensitive( var C : CharSet)
7286:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7287:  Function IntRangeLength( const Low, High : Integer) : Int64
7288:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7289:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7290:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7291:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7292:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7293:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7294:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7295:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7296:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7297:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7298:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7299:   AddTypeS('UnicodeChar', 'WideChar
7300:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7301:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7302:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7303:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7304:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7305:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7306:  Function cSgn( const A : LongInt) : Integer;
7307:  Function cSgn1( const A : Int64) : Integer;
7308:  Function cSgn2( const A : Extended) : Integer;
7309:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7310:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7311:  Function WideCharToInt( const A : WideChar) : Integer
7312:  Function CharToInt( const A : Char) : Integer
7313:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7314:  Function IntToWideChar( const A : Integer) : WideChar
7315:  Function IntToChar( const A : Integer) : Char
7316:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7317:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7318:  Function IsHexChar( const Ch : Char) : Boolean
7319:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7320:  Function HexWideCharToInt( const A : WideChar) : Integer
7321:  Function HexCharToInt( const A : Char) : Integer
7322:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7323:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7324:  Function IntToUpperHexChar( const A : Integer) : Char
7325:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7326:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7327:  Function IntToLowerHexChar( const A : Integer) : Char
7328:  Function IntToStringA( const A : Int64) : AnsiString
```

```
7329:  Function IntToStringW( const A : Int64) : WideString
7330:  Function IntToString( const A : Int64) : String
7331:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7332:  Function UIntToStringW( const A : NativeUInt) : WideString
7333:  Function UIntToString( const A : NativeUInt) : String
7334:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7335:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7336:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7337:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7338:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7339:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7340:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7341:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7342:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7343:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7344:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7345:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7346:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7347:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7348:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7349:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7350:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7351:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7352:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7353:  Function StringToInt64A( const S : AnsiString) : Int64
7354:  Function StringToInt64W( const S : WideString) : Int64
7355:  Function StringToInt64( const S : String) : Int64
7356:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7357:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7358:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7359:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7360:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7361:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7362:  Function StringToIntA( const S : AnsiString) : Integer
7363:  Function StringToIntW( const S : WideString) : Integer
7364:  Function StringToInt( const S : String) : Integer
7365:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7366:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7367:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7368:  Function StringToLongWordA( const S : AnsiString) : LongWord
7369:  Function StringToLongWordW( const S : WideString) : LongWord
7370:  Function StringToLongWord( const S : String) : LongWord
7371:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7372:  Function HexToUIntW( const S : WideString) : NativeUInt
7373:  Function HexToUInt( const S : String) : NativeUInt
7374:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7375:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7376:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7377:  Function HexToLongWordA( const S : AnsiString) : LongWord
7378:  Function HexToLongWordW( const S : WideString) : LongWord
7379:  Function HexToLongWord( const S : String) : LongWord
7380:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7381:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7382:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7383:  Function OctToLongWordA( const S : AnsiString) : LongWord
7384:  Function OctToLongWordW( const S : WideString) : LongWord
7385:  Function OctToLongWord( const S : String) : LongWord
7386:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7387:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7388:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7389:  Function BinToLongWordA( const S : AnsiString) : LongWord
7390:  Function BinToLongWordW( const S : WideString) : LongWord
7391:  Function BinToLongWord( const S : String) : LongWord
7392:  Function FloatToStringA( const A : Extended) : AnsiString
7393:  Function FloatToStringW( const A : Extended) : WideString
7394:  Function FloatToString( const A : Extended) : String
7395:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7396:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7397:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7398:  Function StringToFloatA( const A : AnsiString) : Extended
7399:  Function StringToFloatW( const A : WideString) : Extended
7400:  Function StringToFloat( const A : String) : Extended
7401:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7402:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7403:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7404:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7405:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7406:  unit uPSI_cFundamentUtils;
7407:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7408:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7409:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7410:  Const('CCHARSET','Stringb64_XXEncode);
7411:  Const('CHEXSET','String'0123456789ABCDEF
7412:  Const('HEXDIGITS','String'0123456789ABCDEF
7413:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7414:  Const('DIGISET','String'0123456789
7415:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7416:  Const('DIGISET2','TCharset').SetSet('0123456789'
```

```
7417:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7418:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7419:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7420:  Const('NUMBERS','String'0123456789');
7421:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7422:  Function CharSetToStr( const C : CharSet) : AnsiString
7423:  Function StrToCharSet( const S : AnsiString) : CharSet
7424:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7425:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7426:  Function UUDecode( const S : AnsiString) : AnsiString
7427:  Function XXDecode( const S : AnsiString) : AnsiString
7428:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7429:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7430:  Function InterfaceToStrW( const I : IInterface) : WideString
7431:  Function InterfaceToStr( const I : IInterface) : String
7432:  Function ObjectClassName( const O : TObject) : String
7433:  Function ClassClassName( const C : TClass) : String
7434:  Function ObjectToStr( const O : TObject) : String
7435:  Function ObjectToString( const O : TObject) : String
7436:  Function CharSetToStr( const C : CharSet) : AnsiString
7437:  Function StrToCharSet( const S : AnsiString) : CharSet
7438:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7439:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7440:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7441:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7442:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7443:  Const('Bytes1KB','LongInt'( 1024);
7444:   SIRegister_IInterface(CL);
7445:  Procedure SelfTestCFundamentUtils
7446:
7447: Function CreateSchedule : IJclSchedule
7448: Function NullStamp : TTimeStamp
7449: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7450: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7451: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7452:
7453: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7454: begin
7455:  AddTypeS('TFunc', 'function(X : Float) : Float;
7456: Function InitGraphics( Width, Height : Integer) : Boolean
7457: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7458: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7459: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7460: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7461: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7462: Procedure SetGraphTitle( Title : String)
7463: Procedure SetOxTitle( Title : String)
7464: Procedure SetOyTitle( Title : String)
7465: Function GetGraphTitle : String
7466: Function GetOxTitle : String
7467: Function GetOyTitle : String
7468: Procedure PlotOxAxis( Canvas : TCanvas)
7469: Procedure PlotOyAxis( Canvas : TCanvas)
7470: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7471: Procedure WriteGraphTitle( Canvas : TCanvas)
7472: Function SetMaxCurv( NCurv : Byte) : Boolean
7473: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7474: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7475: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7476: Procedure SetCurvStep( CurvIndex, Step : Integer)
7477: Function GetMaxCurv : Byte
7478: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7479: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7480: Function GetCurvLegend( CurvIndex : Integer) : String
7481: Function GetCurvStep( CurvIndex : Integer) : Integer
7482: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7483: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7484: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7485: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7486: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7487: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7488: Function Xpixel( X : Float) : Integer
7489: Function Ypixel( Y : Float) : Integer
7490: Function Xuser( X : Integer) : Float
7491: Function Yuser( Y : Integer) : Float
7492: end;
7493:
7494: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7495: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7496: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7497: Procedure FFT_Integer_Cleanup
7498: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7499: //unit uPSI_JclStreams;
7500: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7501: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7502: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
```

```
7503: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7504:
7505: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7506: begin
7507:  FindClass('TOBJECT'),'EInvalidDest
7508:  FindClass('TOBJECT'),'EFCantMove
7509:  Procedure fmxCopyFile( const FileName, DestName : string)
7510:  Procedure fmxMoveFile( const FileName, DestName : string)
7511:  Function fmxGetFileSize( const FileName : string) : LongInt
7512:  Function fmxFileDateTime( const FileName : string) : TDateTime
7513:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7514:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7515: end;
7516:
7517: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7518: begin
7519:  SIRegister_IFindFileIterator(CL);
7520:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7521: end;
7522:
7523: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7524: begin
7525:  Function SkipWhite( cp : PChar) : PChar
7526:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7527:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7528:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7529: end;
7530:
7531: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7532: begin
7533:   SIRegister_TStringHashMapTraits(CL);
7534:  Function CaseSensitiveTraits : TStringHashMapTraits
7535:  Function CaseInsensitiveTraits : TStringHashMapTraits
7536:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7537:    +'e; Right : PHashNode; end
7538:   //PHashArray', '^THashArray // will not work
7539:   SIRegister_TStringHashMap(CL);
7540:  THashValue', 'Cardinal
7541:  Function StrHash( const s : string) : THashValue
7542:  Function TextHash( const s : string) : THashValue
7543:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7544:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7545:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7546:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7547:   SIRegister_TCaseSensitiveTraits(CL);
7548:   SIRegister_TCaseInsensitiveTraits(CL);
7549:
7550:
7551: //************************************************************************unit uPSI_umath;
7552:  Function uExpo( X : Float) : Float
7553:  Function uExp2( X : Float) : Float
7554:  Function uExp10( X : Float) : Float
7555:  Function uLog( X : Float) : Float
7556:  Function uLog2( X : Float) : Float
7557:  Function uLog10( X : Float) : Float
7558:  Function uLogA( X, A : Float) : Float
7559:  Function uIntPower( X : Float; N : Integer): Float
7560:  Function uPower( X, Y : Float) : Float
7561:  Function SgnGamma( X : Float) : Integer
7562:  Function Stirling( X : Float) : Float
7563:  Function StirLog( X : Float) : Float
7564:  Function Gamma( X : Float) : Float
7565:  Function LnGamma( X : Float) : Float
7566:  Function DiGamma( X : Float) : Float
7567:  Function TriGamma( X : Float) : Float
7568:  Function IGamma( X : Float) : Float
7569:  Function JGamma( X : Float) : Float
7570:  Function InvGamma( X : Float) : Float
7571:  Function Erf( X : Float) : Float
7572:  Function Erfc( X : Float) : Float
7573: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7574: { Correlation coefficient between samples X and Y }
7575: function DBeta(A, B, X : Float) : Float;
7576: { Density of Beta distribution with parameters A and B }
7577: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7578: Function Beta(X, Y : Float) : Float
7579: Function Binomial( N, K : Integer) : Float
7580: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7581: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7582: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7583: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7584: Function DNorm( X : Float) : Float
7585:
7586: function DGamma(A, B, X : Float) : Float;
7587: { Density of Gamma distribution with parameters A and B }
7588: function DKhi2(Nu : Integer; X : Float) : Float;
7589: { Density of Khi-2 distribution with Nu d.o.f. }
7590: function DStudent(Nu : Integer; X : Float) : Float;
7591: { Density of Student distribution with Nu d.o.f. }
```

```
7592: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7593: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7594: function IBeta(A, B, X : Float) : Float;
7595: { Incomplete Beta function}
7596: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7597:
7598: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7599: begin
7600:  Procedure SetOptAlgo( Algo : TOptAlgo)
7601: procedure SetOptAlgo(Algo : TOptAlgo);
7602: { ----------------------------------------------------------------
7603:   Sets the optimization algorithm according to Algo, which must be
7604:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7605:
7606:  Function GetOptAlgo : TOptAlgo
7607:  Procedure SetMaxParam( N : Byte)
7608:  Function GetMaxParam : Byte
7609:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7610:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7611:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7612:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
      Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7613:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
      MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7614:  Procedure SetMCFile( FileName : String)
7615:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7616:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
      LastPar:Integer;V:TMatrix);
7617: end;
7618:
7619: (*----------------------------------------------------------------------------*)
7620: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7621: begin
7622:  Procedure SaveSimplex( FileName : string)
7623:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7624: end;
7625: (*----------------------------------------------------------------------------*)
7626: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7627: begin
7628:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7629:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7630: end;
7631:
7632: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7633: begin
7634:  Function LTrim( S : String) : String
7635:  Function RTrim( S : String) : String
7636:  Function uTrim( S : String) : String
7637:  Function StrChar( N : Byte; C : Char) : String
7638:  Function RFill( S : String; L : Byte) : String
7639:  Function LFill( S : String; L : Byte) : String
7640:  Function CFill( S : String; L : Byte) : String
7641:  Function Replace( S : String; C1, C2 : Char) : String
7642:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7643:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7644:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7645:  Function FloatStr( X : Float) : String
7646:  Function IntStr( N : LongInt) : String
7647:  Function uCompStr( Z : Complex) : String
7648: end;
7649:
7650: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7651: begin
7652:  Function uSinh( X : Float) : Float
7653:  Function uCosh( X : Float) : Float
7654:  Function uTanh( X : Float) : Float
7655:  Function uArcSinh( X : Float) : Float
7656:  Function uArcCosh( X : Float) : Float
7657:  Function ArcTanh( X : Float) : Float
7658:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7659: end;
7660:
7661: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7662: begin
7663: type RNG_Type =
7664:   (RNG_MWC,      { Multiply-With-Carry }
7665:    RNG_MT,       { Mersenne Twister }
7666:    RNG_UVAG);    { Universal Virtual Array Generator }
7667:  Procedure SetRNG( RNG : RNG_Type)
7668:  Procedure InitGen( Seed : RNG_IntType)
7669:  Procedure SRand( Seed : RNG_IntType)
7670:  Function IRanGen : RNG_IntType
7671:  Function IRanGen31 : RNG_IntType
7672:  Function RanGen1 : Float
7673:  Function RanGen2 : Float
7674:  Function RanGen3 : Float
7675:  Function RanGen53 : Float
7676: end;
7677:
```

```
7678: //  Optimization by Simulated Annealing
7679: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7680: begin
7681:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7682:  Procedure SA_CreateLogFile( FileName : String)
7683:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7684: end;
7685:
7686: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7687: begin
7688:  Procedure InitUVAGbyString( KeyPhrase : string)
7689:  Procedure InitUVAG( Seed : RNG_IntType)
7690:  Function IRanUVAG : RNG_IntType
7691: end;
7692:
7693: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7694: begin
7695:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7696:  Procedure GA_CreateLogFile( LogFileName : String)
7697:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7698: end;
7699:
7700:  TVector', 'array of Float
7701: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7702: begin
7703:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7704:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7705: end;
7706:
7707: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7708: begin
7709:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7710:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7711: end;
7712:
7713: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7714: begin
7715:   FT_Result', 'Integer
7716:   //TDWordptr', '^DWord // will not work
7717:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7718:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7719:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7720:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7721:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7722:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7723:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7724:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7725:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7726:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7727:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7728:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7729:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7730:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7731:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7732:    yte; end
7733: end;
7734:
7735:
7736: //************************************ PaintFX*****************************
7737: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7738: begin
7739:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7740:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7741:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7742:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7743:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7744:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7745:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7746:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7747:     Procedure Turn( Src, Dst : TBitmap)
7748:     Procedure TurnRight( Src, Dst : TBitmap)
7749:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7750:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7751:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7752:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7753:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7754:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7755:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7756:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7757:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7758:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7759:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7760:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7761:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7762:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7763:     Procedure Emboss( var Bmp : TBitmap)
7764:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7765:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7766:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
```

```
7767:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7768:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7769:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7770:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7771:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7772:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7773:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7774:     Procedure SemiOpaque( Src, Dst : TBitmap)
7775:     Procedure ShadowDownLeft( const Dst : TBitmap)
7776:     Procedure ShadowDownRight( const Dst : TBitmap)
7777:     Procedure ShadowUpLeft( const Dst : TBitmap)
7778:     Procedure ShadowUpRight( const Dst : TBitmap)
7779:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7780:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7781:     Procedure FlipRight( const Dst : TBitmap)
7782:     Procedure FlipDown( const Dst : TBitmap)
7783:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7784:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7785:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7786:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7787:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7788:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7789:     Procedure SmoothResize( var Src, Dst : TBitmap)
7790:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7791:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7792:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7793:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7794:     Procedure GrayScale( const Dst : TBitmap)
7795:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7796:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7797:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7798:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7799:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7800:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7801:     Procedure AntiAlias( const Dst : TBitmap)
7802:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7803:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7804:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7805:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7806:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7807:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7808:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7809:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7810:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7811:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7812:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7813:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7814:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7815:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7816:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7817:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7818:     Procedure Invert( Src : TBitmap)
7819:     Procedure MirrorRight( Src : TBitmap)
7820:     Procedure MirrorDown( Src : TBitmap)
7821:   end;
7822: end;
7823:
7824: (*----------------------------------------------------------------------------*)
7825: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7826: begin
7827:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7828:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7829:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7830:   SIRegister_TJvPaintFX(CL);
7831:  Function SplineFilter( Value : Single) : Single
7832:  Function BellFilter( Value : Single) : Single
7833:  Function TriangleFilter( Value : Single) : Single
7834:  Function BoxFilter( Value : Single) : Single
7835:  Function HermiteFilter( Value : Single) : Single
7836:  Function Lanczos3Filter( Value : Single) : Single
7837:  Function MitchellFilter( Value : Single) : Single
7838: end;
7839:
7840:
7841: (*----------------------------------------------------------------------------*)
7842: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7843: begin
7844:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7845:  TeeMsg_DefaultSeriesName','String 'Series
7846:  TeeMsg_DefaultToolName','String 'ChartTool
7847:  ChartComponentPalette','String 'TeeChart
7848:  TeeMaxLegendColumns',LongInt'( 2);
7849:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7850:  TeeTitleFootDistance,LongInt( 5);
7851:   SIRegister_TCustomChartWall(CL);
7852:   SIRegister_TChartWall(CL);
7853:   SIRegister_TChartLegendGradient(CL);
7854:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7855:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
```

```
7856:   FindClass('TOBJECT'),'LegendException
7857:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7858:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7859:   FindClass('TOBJECT'),'TCustomChartLegend
7860:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7861:   TLegendSymbolPosition', '( spLeft, spRight )
7862:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7863:   TSymbolCalcHeight', 'Function  : Integer
7864:   SIRegister_TLegendSymbol(CL);
7865:   SIRegister_TTeeCustomShapePosition(CL);
7866:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7867:   SIRegister_TLegendTitle(CL);
7868:   SIRegister_TLegendItem(CL);
7869:   SIRegister_TLegendItems(CL);
7870:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7871:   FindClass('TOBJECT'),'TCustomChart
7872:   SIRegister_TCustomChartLegend(CL);
7873:   SIRegister_TChartLegend(CL);
7874:   SIRegister_TChartTitle(CL);
7875:   SIRegister_TChartFootTitle(CL);
7876:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7877:    +'eButton; Shift : TShiftState; X, Y : Integer)
7878:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7879:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7880:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7881:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7882:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7883:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7884:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7885:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7886:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7887:    +'toMax : Boolean; Min : Double; Max : Double; end
7888:   TAllAxisSavedScales', 'array of TAxisSavedScales
7889:   SIRegister_TChartBackWall(CL);
7890:   SIRegister_TChartRightWall(CL);
7891:   SIRegister_TChartBottomWall(CL);
7892:   SIRegister_TChartLeftWall(CL);
7893:   SIRegister_TChartWalls(CL);
7894:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7895:   SIRegister_TCustomChart(CL);
7896:   SIRegister_TChart(CL);
7897:   SIRegister_TTeeSeriesTypes(CL);
7898:   SIRegister_TTeeToolTypes(CL);
7899:   SIRegister_TTeeDragObject(CL);
7900:   SIRegister_TColorPalettes(CL);
7901:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7902:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7903:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7904:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7905:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7906:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7907:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7908:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7909:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7910:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7911:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7912:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7913:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7914:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7915:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7916:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7917:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7918:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7919:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7920:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7921:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7922:   SIRegister_TChartTheme(CL);
7923:   //TChartThemeClass', 'class of TChartTheme
7924:   //TCanvasClass', 'class of TCanvas3D
7925:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7926:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7927:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7928:  Procedure ShowMessageUser( const S : String)
7929:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7930:  Function HasLabels( ASeries : TChartSeries) : Boolean
7931:  Function HasColors( ASeries : TChartSeries) : Boolean
7932:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7933:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7934: end;
7935:
7936:
7937: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7938: begin
7939:  //'TeeFormBorderStyle',' bsNone);
```

```
7940:   SIRegister_TMetafile(CL);
7941:  'TeeDefVerticalMargin','LongInt'( 4);
7942:  'TeeDefHorizMargin','LongInt'( 3);
7943:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7944:  'TeeMsg_TeeHand','String 'crTeeHand
7945:  'TeeNormalPrintDetail','LongInt'( 0);
7946:  'TeeHighPrintDetail','LongInt'( - 100);
7947:  'TeeDefault_PrintMargin','LongInt'( 15);
7948:  'MaxDefaultColors','LongInt'( 19);
7949:  'TeeTabDelimiter','Char #9);
7950:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7951:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7952:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7953:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7954:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7955:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7956:   SIRegister_TCustomPanelNoCaption(CL);
7957:   FindClass('TOBJECT'),'TCustomTeePanel
7958:   SIRegister_TZoomPanning(CL);
7959:   SIRegister_TTeeEvent(CL);
7960:   //SIRegister_TTeeEventListeners(CL);
7961:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7962:   SIRegister_TTeeMouseEvent(CL);
7963:   SIRegister_TCustomTeePanel(CL);
7964:   //TChartGradient', 'TTeeGradient
7965:   //TChartGradientClass', 'class of TChartGradient
7966:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7967:   SIRegister_TTeeZoomPen(CL);
7968:   SIRegister_TTeeZoomBrush(CL);
7969:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7970:   SIRegister_TTeeZoom(CL);
7971:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7972:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7973:   SIRegister_TBackImage(CL);
7974:   SIRegister_TCustomTeePanelExtended(CL);
7975:   //TChartBrushClass', 'class of TChartBrush
7976:   SIRegister_TTeeCustomShapeBrushPen(CL);
7977:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7978:   TTextFormat', '( ttfNormal, ttfHtml )
7979:   SIRegister_TTeeCustomShape(CL);
7980:   SIRegister_TTeeShape(CL);
7981:   SIRegister_TTeeExportData(CL);
7982:  Function TeeStr( const Num : Integer) : String
7983:  Function DateTimeDefaultFormat( const AStep : Double) : String
7984:  Function TEEDaysInMonth( Year, Month : Word) : Word
7985:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7986:  Function NextDateTimeStep( const AStep : Double) : Double
7987:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7988:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7989:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7990:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7991:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7992:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7993:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7994:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7995:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7996:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7997:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7998:  Function DelphiToLocalFormat( const Format : String) : String
7999:  Function LocalToDelphiFormat( const Format : String) : String
8000:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8001:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8002:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8003:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
8004:   TTeeSortSwap', 'Procedure ( a, b : Integer)
8005:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8006:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
8007:  Function TeeExtractField( St : String; Index : Integer) : String;
8008:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
8009:  Function TeeNumFields( St : String) : Integer;
8010:  Function TeeNumFields1( const St, Separator : String) : Integer;
8011:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8012:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8013:  // TColorArray', 'array of TColor
8014:  Function GetDefaultColor( const Index : Integer) : TColor
8015:  Procedure SetDefaultColorPalette;
8016:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
8017:  'TeeCheckBoxSize','LongInt'( 11);
8018:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8019:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
8020:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8021:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8022:  Procedure TeeTranslateControl( AControl : TControl);
8023:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
8024:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8025:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8026:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8027:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
```

```
8028:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8029:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8030:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8031:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8032:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8033:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8034:  Procedure TeeSaveStringOption( const AKey, Value : String)
8035:  Function TeeDefaultXMLEncoding : String
8036:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8037:   TeeWindowHandle', 'Integer
8038:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8039:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8040:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8041: end;
8042:
8043:
8044: using mXBDEUtils
8045: ********************************************************************************
8046:  Procedure SetAlias( aAlias, aDirectory : String)
8047:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
8048:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8049:  Procedure SetBDE( aPath, aNode, aValue : String)
8050:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8051:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8052:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8053:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8054:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8055:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8056:
8057:
8058: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8059: begin
8060: AddClassN(FindClass('TOBJECT'),'EDateTime
8061: Function DatePart( const D : TDateTime) : Integer
8062: Function TimePart( const D : TDateTime) : Double
8063: Function Century( const D : TDateTime) : Word
8064: Function Year( const D : TDateTime) : Word
8065: Function Month( const D : TDateTime) : Word
8066: Function Day( const D : TDateTime) : Word
8067: Function Hour( const D : TDateTime) : Word
8068: Function Minute( const D : TDateTime) : Word
8069: Function Second( const D : TDateTime) : Word
8070: Function Millisecond( const D : TDateTime) : Word
8071: ('OneDay','Extended').setExtended( 1.0);
8072: ('OneHour','Extended').SetExtended( OneDay / 24);
8073: ('OneMinute','Extended').SetExtended( OneHour / 60);
8074: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8075: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8076: ('OneWeek','Extended').SetExtended( OneDay * 7);
8077: ('HoursPerDay','Extended').SetExtended( 24);
8078: ('MinutesPerHour','Extended').SetExtended( 60);
8079: ('SecondsPerMinute','Extended').SetExtended( 60);
8080: Procedure SetYear( var D : TDateTime; const Year : Word)
8081: Procedure SetMonth( var D : TDateTime; const Month : Word)
8082: Procedure SetDay( var D : TDateTime; const Day : Word)
8083: Procedure SetHour( var D : TDateTime; const Hour : Word)
8084: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8085: Procedure SetSecond( var D : TDateTime; const Second : Word)
8086: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8087: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8088: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8089: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8090: Function IsAM( const D : TDateTime) : Boolean
8091: Function IsPM( const D : TDateTime) : Boolean
8092: Function IsMidnight( const D : TDateTime) : Boolean
8093: Function IsNoon( const D : TDateTime) : Boolean
8094: Function IsSunday( const D : TDateTime) : Boolean
8095: Function IsMonday( const D : TDateTime) : Boolean
8096: Function IsTuesday( const D : TDateTime) : Boolean
8097: Function IsWednesday( const D : TDateTime) : Boolean
8098: Function IsThursday( const D : TDateTime) : Boolean
8099: Function IsFriday( const D : TDateTime) : Boolean
8100: Function IsSaturday( const D : TDateTime) : Boolean
8101: Function IsWeekend( const D : TDateTime) : Boolean
8102: Function Noon( const D : TDateTime) : TDateTime
8103: Function Midnight( const D : TDateTime) : TDateTime
8104: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8105: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8106: Function NextWorkday( const D : TDateTime) : TDateTime
8107: Function PreviousWorkday( const D : TDateTime) : TDateTime
8108: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8109: Function LastDayOfYear( const D : TDateTime) : TDateTime
8110: Function EasterSunday( const Year : Word) : TDateTime
8111: Function GoodFriday( const Year : Word) : TDateTime
8112: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8113: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8114: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8115: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
```

```
8116: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8117: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8118: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8119: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8120: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8121: Function DayOfYear( const D : TDateTime) : Integer
8122: Function DaysInMonth( const Ye, Mo : Word) : Integer
8123: Function DaysInMonth( const D : TDateTime) : Integer
8124: Function DaysInYear( const Ye : Word) : Integer
8125: Function DaysInYearDate( const D : TDateTime) : Integer
8126: Function WeekNumber( const D : TDateTime) : Integer
8127: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8128: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8129: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8130: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8131: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8132: Function DiffHours( const D1, D2 : TDateTime) : Integer
8133: Function DiffDays( const D1, D2 : TDateTime) : Integer
8134: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8135: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8136: Function DiffYears( const D1, D2 : TDateTime) : Integer
8137: Function GMTBias : Integer
8138: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8139: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8140: Function NowAsGMTTime : TDateTime
8141: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8142: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8143: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8144: Function DateTimeToANSI( const D : TDateTime) : Integer
8145: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8146: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8147: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8148: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8149: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8150: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8151: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8152: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8153: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8154: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8155: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8156: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8157: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8158: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8159: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8160: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8161: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8162: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8163: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8164: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8165: Function EnglishShortMonthA( const S : AnsiString) : Integer
8166: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8167: Function EnglishLongMonthA( const S : AnsiString) : Integer
8168: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8169: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8170: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8171: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8172: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8173: Function RFCMonthA( const S : AnsiString) : Word
8174: Function RFCMonthU( const S : UnicodeString) : Word
8175: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8176: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8177: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8178: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8179: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8180: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8181: Function NowAsRFCDateTimeA : AnsiString
8182: Function NowAsRFCDateTimeU : UnicodeString
8183: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8184: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8185: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8186: Function TimePeriodStr( const D : TDateTime) : AnsiString
8187: Procedure SelfTest
8188: end;
8189: //*****************************************CFileUtils
8190: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8191: Function PathHasDriveLetter( const Path : String) : Boolean
8192: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8193: Function PathIsDriveLetter( const Path : String) : Boolean
8194: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8195: Function PathIsDriveRoot( const Path : String) : Boolean
8196: Function PathIsRootA( const Path : AnsiString) : Boolean
8197: Function PathIsRoot( const Path : String) : Boolean
8198: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8199: Function PathIsUNCPath( const Path : String) : Boolean
8200: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8201: Function PathIsAbsolute( const Path : String) : Boolean
8202: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8203: Function PathIsDirectory( const Path : String) : Boolean
8204: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
```

```
8205: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8206: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8207: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8208: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8209: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8210: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8211: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8212: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8213: Function PathCanonical( const Path : String; const PathSep : Char) : String
8214: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8215: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8216: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8217: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8218: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8219: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8220: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8221: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8222: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8223: Function FileNameValid( const FileName : String) : String
8224: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8225: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8226: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8227: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8228: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8229: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8230: Procedure CCopyFile( const FileName, DestName : String)
8231: Procedure CMoveFile( const FileName, DestName : String)
8232: Function CDeleteFiles( const FileMask : String) : Boolean
8233: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8234: Procedure FileCloseEx( const FileHandle : TFileHandle)
8235: Function FileExistsA( const FileName : AnsiString) : Boolean
8236:  Function CFileExists( const FileName : String) : Boolean
8237:  Function CFileGetSize( const FileName : String) : Int64
8238: Function FileGetDateTime( const FileName : String) : TDateTime
8239: Function FileGetDateTime2( const FileName : String) : TDateTime
8240: Function FileIsReadOnly( const FileName : String) : Boolean
8241: Procedure FileDeleteEx( const FileName : String)
8242: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8243: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
     : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8244: Function DirectoryEntryExists( const Name : String) : Boolean
8245: Function DirectoryEntrySize( const Name : String) : Int64
8246:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8247: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8248: Procedure CDirectoryCreate( const DirectoryName : String)
8249: Function GetFirstFileNameMatching( const FileMask : String) : String
8250: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8251: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8252: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8253:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8254:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8255: Function DriveIsValid( const Drive : Char) : Boolean
8256: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8257: Function DriveFreeSpace( const Path : AnsiString) : Int64
8258:
8259: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8260: begin
8261:  AddClassN(FindClass('TOBJECT'),'ETimers
8262:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8263: Function TickDelta( const D1, D2 : LongWord) : Integer
8264: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8265:   AddTypeS('THPTimer', 'Int64
8266: Procedure StartTimer( var Timer : THPTimer)
8267: Procedure StopTimer( var Timer : THPTimer)
8268: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8269: Procedure InitStoppedTimer( var Timer : THPTimer)
8270: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8271: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8272: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8273: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8274: Function GetHighPrecisionFrequency : Int64
8275: Function GetHighPrecisionTimerOverhead : Int64
8276: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8277: Procedure SelfTestCTimer
8278: end;
8279:
8280: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8281: begin
8282:  Function RandomSeed : LongWord
8283:  Procedure AddEntropy( const Value : LongWord)
8284:  Function RandomUniform : LongWord;
8285:  Function RandomUniform1( const N : Integer) : Integer;
8286:  Function RandomBoolean : Boolean
8287:  Function RandomByte : Byte
8288:  Function RandomByteNonZero : Byte
8289:  Function RandomWord : Word
8290:  Function RandomInt64 : Int64;
8291:  Function RandomInt641( const N : Int64) : Int64;
8292:  Function RandomHex( const Digits : Integer) : String
```

```
8293:  Function RandomFloat : Extended
8294:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8295:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8296:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8297:  Function mwcRandomLongWord : LongWord
8298:  Function urnRandomLongWord : LongWord
8299:  Function moaRandomFloat : Extended
8300:  Function mwcRandomFloat : Extended
8301:  Function RandomNormalF : Extended
8302:  Procedure SelfTestCRandom
8303: end;
8304:
8305: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8306: begin
8307:  // PIntArray', '^TIntArray // will not work
8308:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8309:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8310:  Function synMax( x, y : integer) : integer
8311:  Function synMin( x, y : integer) : integer
8312:  Function synMinMax( x, mi, ma : integer) : integer
8313:  Procedure synSwapInt( var l, r : integer)
8314:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8315:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8316:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8317:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8318:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8319:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8320:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8321:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8322:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8323:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8324:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8325:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8326:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8327:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8328:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8329:  ('C3_NONSPACING','LongInt'( 1);
8330:  'C3_DIACRITIC','LongInt'( 2);
8331:  'C3_VOWELMARK','LongInt'( 4);
8332:  ('C3_SYMBOL','LongInt'( 8);
8333:  ('C3_KATAKANA','LongWord( $0010);
8334:  ('C3_HIRAGANA','LongWord( $0020);
8335:  ('C3_HALFWIDTH','LongWord( $0040);
8336:  ('C3_FULLWIDTH','LongWord( $0080);
8337:  ('C3_IDEOGRAPH','LongWord( $0100);
8338:  ('C3_KASHIDA','LongWord( $0200);
8339:  ('C3_LEXICAL','LongWord( $0400);
8340:  ('C3_ALPHA','LongWord( $8000);
8341:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8342:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8343:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8344:  Function synIsStringType( Value : Word) : TStringType
8345:  Function synGetEOL( Line : PChar) : PChar
8346:  Function synEncodeString( s : string) : string
8347:  Function synDecodeString( s : string) : string
8348:  Procedure synFreeAndNil( var Obj: TObject)
8349:  Procedure synAssert( Expr : Boolean)
8350:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8351:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8352:   TReplaceFlags', 'set of TReplaceFlag )
8353:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8354:  Function synGetRValue( RGBValue : TColor) : byte
8355:  Function synGetGValue( RGBValue : TColor) : byte
8356:  Function synGetBValue( RGBValue : TColor) : byte
8357:  Function synRGB( r, g, b : Byte) : Cardinal
8358: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8359:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8360:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8361:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8362:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8363: end;
8364:
8365:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8366:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8367:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8368:
8369: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8370: begin
8371:  Function STimeZoneBias : integer
8372:  Function TimeZone : string
8373:  Function Rfc822DateTime( t : TDateTime) : string
8374:  Function CDateTime( t : TDateTime) : string
8375:  Function SimpleDateTime( t : TDateTime) : string
8376:  Function AnsiCDateTime( t : TDateTime) : string
8377:  Function GetMonthNumber( Value : String) : integer
8378:  Function GetTimeFromStr( Value : string) : TDateTime
8379:  Function GetDateMDYFromStr( Value : string) : TDateTime
```

```
8380:  Function DecodeRfcDateTime( Value : string) : TDateTime
8381:  Function GetUTTime : TDateTime
8382:  Function SetUTTime( Newdt : TDateTime) : Boolean
8383:  Function SGetTick : LongWord
8384:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8385:  Function CodeInt( Value : Word) : Ansistring
8386:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8387:  Function CodeLongInt( Value : LongInt) : Ansistring
8388:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8389:  Function DumpStr( const Buffer : Ansistring) : string
8390:  Function DumpExStr( const Buffer : Ansistring) : string
8391:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8392:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8393:  Function TrimSPLeft( const S : string) : string
8394:  Function TrimSPRight( const S : string) : string
8395:  Function TrimSP( const S : string) : string
8396:  Function SeparateLeft( const Value, Delimiter : string) : string
8397:  Function SeparateRight( const Value, Delimiter : string) : string
8398:  Function SGetParameter( const Value, Parameter : string) : string
8399:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8400:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8401:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8402:  Function GetEmailAddr( const Value : string) : string
8403:  Function GetEmailDesc( Value : string) : string
8404:  Function CStrToHex( const Value : Ansistring) : string
8405:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8406:  Function CBinToInt( const Value : string) : Integer
8407:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8408:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8409:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8410:  Function CRPos( const Sub, Value : String) : Integer
8411:  Function FetchBin( var Value : string; const Delimiter : string) : string
8412:  Function CFetch( var Value : string; const Delimiter : string) : string
8413:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8414:  Function IsBinaryString( const Value : AnsiString) : Boolean
8415:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8416:  Procedure StringsTrim( const value : TStrings)
8417:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8418:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8419:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8420:  Function CCountOfChar( const Value : string; aChr : char) : integer
8421:  Function UnquoteStr( const Value : string; Quote : Char) : string
8422:  Function QuoteStr( const Value : string; Quote : Char) : string
8423:  Procedure HeadersToList( const Value : TStrings)
8424:  Procedure ListToHeaders( const Value : TStrings)
8425:  Function SwapBytes( Value : integer) : integer
8426:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8427:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8428:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8429:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8430:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8431:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8432: end;
8433:
8434: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8435: begin
8436:  ('CrcBufSize','LongInt'( 2048);
8437:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8438:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8439:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8440:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8441:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8442:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8443:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8444:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8445:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8446:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8447:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8448:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8449:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8450:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8451:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8452: end;
8453:
8454: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8455: begin
8456:  function CreateOleObject(const ClassName: String): IDispatch;
8457:  function GetActiveOleObject(const ClassName: String): IDispatch;
8458:  function ProgIDToClassID(const ProgID: string): TGUID;
8459:  function ClassIDToProgID(const ClassID: TGUID): string;
8460:  function CreateClassID: string;
8461:  function CreateGUIDString: string;
8462:  function CreateGUIDID: string;
8463:  procedure OleError(ErrorCode: longint)
8464:  procedure OleCheck(Result: HResult);
8465: end;
8466:
8467:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8468:  Function xGetActiveOleObject( const ClassName : string) : Variant
```

```
8469:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8470:  Function DllCanUnloadNow : HResult
8471:  Function DllRegisterServer : HResult
8472:  Function DllUnregisterServer : HResult
8473:  Function VarFromInterface( Unknown : IUnknown) : Variant
8474:  Function VarToInterface( const V : Variant) : IDispatch
8475:  Function VarToAutoObject( const V : Variant) : TAutoObject
8476:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8477:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8478:  Procedure OleError( ErrorCode : HResult)
8479:  Procedure OleCheck( Result : HResult)
8480:  Function StringToClassID( const S : string) : TCLSID
8481:  Function ClassIDToString( const ClassID : TCLSID) : string
8482:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8483:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8484:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8485:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8486:  Function xGUIDToString( const ClassID : TGUID) : string
8487:  Function xStringToGUID( const S : string) : TGUID
8488:  Function xGetModuleName( Module : HMODULE) : string
8489:  Function xAcquireExceptionObject : TObject
8490:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8491:  Function xUtf8Encode( const WS : WideString) : UTF8String
8492:  Function xUtf8Decode( const S : UTF8String) : WideString
8493:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8494:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8495:  Function XRTLHandleCOMException : HResult
8496:  Procedure XRTLCheckArgument( Flag : Boolean)
8497:  //Procedure XRTLCheckOutArgument( out Arg)
8498:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8499:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8500:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8501:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8502:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8503:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8504: function   XRTLDefaultCategoryManager: IUnknown;
8505: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8506: // ICatRegister helper functions
8507: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8508:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8509:                                        const CategoryManager: IUnknown = nil): HResult;
8510: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8511:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8512:                                        const CategoryManager: IUnknown = nil): HResult;
8513: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8514:                                        const CategoryManager: IUnknown = nil): HResult;
8515: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8516:                                          const CategoryManager: IUnknown = nil): HResult;
8517: // ICatInformation helper functions
8518: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8519:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8520:                                       const CategoryManager: IUnknown = nil): HResult;
8521: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8522:                                const CategoryManager: IUnknown = nil): HResult;
8523: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8524:                                     const CategoryManager: IUnknown = nil): HResult;
8525: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8526:                                       const CategoryManager: IUnknown = nil): HResult;
8527: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8528:                    const ADelete: Boolean = True): WideString;
8529: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8530: Function XRTLGetVariantAsString( const Value : Variant) : string
8531: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8532: Function XRTLGetTimeZones : TXRTLTimeZones
8533: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8534: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8535: Function GMTNow : TDateTime
8536: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8537: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8538: Procedure XRTLNotImplemented
8539: Procedure XRTLRaiseError( E : Exception)
8540: Procedure XRTLRaise( E : Exception)');
8541: Procedure XRaise( E : Exception)');
8542: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8543:
8544:
8545: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8546: begin
8547:   SIRegister_IXRTLValue(CL);
8548:   SIRegister_TXRTLValue(CL);
8549:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8550:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8551: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8552: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8553: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8554: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
```

```
8555: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8556: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8557: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8558: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8559: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8560: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8561: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8562: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8563: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8564: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8565: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8566: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8567: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8568: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8569: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8570: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8571: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8572: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8573: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8574: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8575: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8576: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8577: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8578:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8579: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8580: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8581: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8582: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8583: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8584: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8585: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8586: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8587: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8588: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8589: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8590: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8591: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8592: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8593: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8594: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8595: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8596: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8597: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8598: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8599: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8600: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8601: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8602: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8603: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8604: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8605: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8606: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8607: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8608: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8609: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8610: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8611: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8612: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8613: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8614: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8615: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8616: end;
8617:
8618: //*****************************unit uPSI_GR32;****************************************
8619:
8620:  Function Color32( WinColor : TColor) : TColor32;
8621:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8622:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8623:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8624:  Function WinColor( Color32 : TColor32) : TColor
8625:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8626:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8627:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8628:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8629:  Function RedComponent( Color32 : TColor32) : Integer
8630:  Function GreenComponent( Color32 : TColor32) : Integer
8631:  Function BlueComponent( Color32 : TColor32) : Integer
8632:  Function AlphaComponent( Color32 : TColor32) : Integer
8633:  Function Intensity( Color32 : TColor32) : Integer
8634:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8635:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8636:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8637:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8638:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8639:  Function WinPalette( const P : TPalette32) : HPALETTE
8640:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8641:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8642:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
```

```
8643:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8644:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8645:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8646:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8647:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8648:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8649:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8650:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8651:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8652:  Function FixedRect1( const ARect : TRect) : TRect;
8653:  Function FixedRect2( const FR : TFloatRect) : TRect;
8654:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8655:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8656:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8657:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8658:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8659:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8660:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8661:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8662:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8663:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8664:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8665:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8666:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8667:  Function IsRectEmpty( const R : TRect) : Boolean;
8668:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8669:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8670:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8671:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8672:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8673:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8674:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8675:  Function MessageBeep( uType : UINT) : BOOL
8676:  Function ShowCursor( bShow : BOOL) : Integer
8677:  Function SetCursorPos( X, Y : Integer) : BOOL
8678:  Function SetCursor( hCursor : HICON) : HCURSOR
8679:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8680:  //Function ClipCursor( lpRect : PRect) : BOOL
8681:  Function GetClipCursor( var lpRect : TRect) : BOOL
8682:  Function GetCursor : HCURSOR
8683:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8684:  Function GetCaretBlinkTime : UINT
8685:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8686:  Function DestroyCaret : BOOL
8687:  Function HideCaret( hWnd : HWND) : BOOL
8688:  Function ShowCaret( hWnd : HWND) : BOOL
8689:  Function SetCaretPos( X, Y : Integer) : BOOL
8690:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8691:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8692:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8693:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8694:  Function WindowFromPoint( Point : TPoint) : HWND
8695:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8696:
8697:
8698: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8699: begin
8700:  Function FixedFloor( A : TFixed) : Integer
8701:  Function FixedCeil( A : TFixed) : Integer
8702:  Function FixedMul( A, B : TFixed) : TFixed
8703:  Function FixedDiv( A, B : TFixed) : TFixed
8704:  Function OneOver( Value : TFixed) : TFixed
8705:  Function FixedRound( A : TFixed) : Integer
8706:  Function FixedSqr( Value : TFixed) : TFixed
8707:  Function FixedSqrtLP( Value : TFixed) : TFixed
8708:  Function FixedSqrtHP( Value : TFixed) : TFixed
8709:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8710:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8711:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8712:  Function GRHypot( const X, Y : TFloat) : TFloat;
8713:  Function Hypot1( const X, Y : Integer) : Integer;
8714:  Function FastSqrt( const Value : TFloat) : TFloat
8715:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8716:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8717:  Function FastInvSqrt( const Value : Single) : Single;
8718:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8719:  Function GRIsPowerOf2( Value : Integer) : Boolean
8720:  Function PrevPowerOf2( Value : Integer) : Integer
8721:  Function NextPowerOf2( Value : Integer) : Integer
8722:  Function Average( A, B : Integer) : Integer
8723:  Function GRSign( Value : Integer) : Integer
8724:  Function FloatMod( x, y : Double) : Double
8725: end;
8726:
8727: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8728: begin
8729:  Function Clamp( const Value : Integer) : Integer;
8730:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8731:  Function StackAlloc( Size : Integer) : Pointer
```

```
8732:  Procedure StackFree( P : Pointer)
8733:  Procedure Swap( var A, B : Pointer);
8734:  Procedure Swap1( var A, B : Integer);
8735:  Procedure Swap2( var A, B : TFixed);
8736:  Procedure Swap3( var A, B : TColor32);
8737:  Procedure TestSwap( var A, B : Integer);
8738:  Procedure TestSwap1( var A, B : TFixed);
8739:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8740:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8741:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8742:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8743:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8744:  Function GRMin( const A, B, C : Integer) : Integer;
8745:  Function GRMax( const A, B, C : Integer) : Integer;
8746:  Function Clamp( Value, Max : Integer) : Integer;
8747:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8748:  Function Wrap( Value, Max : Integer) : Integer;
8749:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8750:  Function Wrap3( Value, Max : Single) : Single;;
8751:  Function WrapPow2( Value, Max : Integer) : Integer;
8752:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8753:  Function Mirror( Value, Max : Integer) : Integer;
8754:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8755:  Function MirrorPow2( Value, Max : Integer) : Integer;
8756:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8757:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8758:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8759:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8760:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8761:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8762:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8763:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8764:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8765:  Function Div255( Value : Cardinal) : Cardinal
8766:  Function SAR_4( Value : Integer) : Integer
8767:  Function SAR_8( Value : Integer) : Integer
8768:  Function SAR_9( Value : Integer) : Integer
8769:  Function SAR_11( Value : Integer) : Integer
8770:  Function SAR_12( Value : Integer) : Integer
8771:  Function SAR_13( Value : Integer) : Integer
8772:  Function SAR_14( Value : Integer) : Integer
8773:  Function SAR_15( Value : Integer) : Integer
8774:  Function SAR_16( Value : Integer) : Integer
8775:  Function ColorSwap( WinColor : TColor) : TColor32
8776: end;
8777:
8778: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8779: begin
8780:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8781:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8782:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8783:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8784:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8785:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8786:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8787:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8788:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8789:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8790:  Function CreateBitmask( Components : TColor32Components) : TColor32
8791:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8792:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8793:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8794: end;
8795:
8796:
8797: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8798: begin
8799:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8800:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8801:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8802:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8803:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8804:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8805:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8806:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8807:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8808:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8809:    //+'tedRangeBuffer; MoreData : Boolean; end
8810:  Function NtfsSetSparse( const FileName : string) : Boolean
8811:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8812:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8813:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8814:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8815:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
```

```
8816:  Function NtfsGetSparse( const FileName : string) : Boolean
8817:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8818:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8819:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8820:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8821:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8822:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8823:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8824:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8825:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8826:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8827:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8828:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8829:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8830:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8831:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8832:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8833:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8834:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8835:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8836:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8837:   AddTypeS('TStreamIds', 'set of TStreamId
8838:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8839:    +': ___Pointer; StreamIds : TStreamIds; end
8840:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8841:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8842:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8843:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8844:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8845:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8846:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8847:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8848:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
     List:TStrings):Bool;
8849:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8850:  Function JclAppInstances : TJclAppInstances;
8851:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8852:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8853:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8854:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8855:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8856:
8857:
8858: (*----------------------------------------------------------------------------*)
8859: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8860: begin
8861:   FindClass('TOBJECT'),'EJclGraphicsError
8862:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8863:   TDynPointArray', 'array of TPoint
8864:   TDynDynPointArrayArray', 'array of TDynPointArray
8865:   TPointF', 'record X : Single; Y : Single; end
8866:   TDynPointArrayF', 'array of TPointF
8867:   TDrawMode2', '( dmOpaque, dmBlend )
8868:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8869:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8870:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8871:   TMatrix3d', 'record array[0..2,0..2] of extended end
8872:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8873:   TScanLine', 'array of Integer
8874:   TScanLines', 'array of TScanLine
8875:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8876:   TGradientDirection', '( gdVertical, gdHorizontal )
8877:   TPolyFillMode', '( fmAlternate, fmWinding )
8878:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8879:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8880:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8881:   SIRegister_TJclDesktopCanvas(CL);
8882:   FindClass('TOBJECT'),'TJclRegion
8883:   SIRegister_TJclRegionInfo(CL);
8884:   SIRegister_TJclRegion(CL);
8885:   SIRegister_TJclThreadPersistent(CL);
8886:   SIRegister_TJclCustomMap(CL);
8887:   SIRegister_TJclBitmap32(CL);
8888:   SIRegister_TJclByteMap(CL);
8889:   SIRegister_TJclTransformation(CL);
8890:   SIRegister_TJclLinearTransformation(CL);
8891:  Procedure Stretch(NewWidth,
     NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8892:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8893:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8894:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8895:  Procedure BitmapToJPeg( const FileName : string)
8896:  Procedure JPegToBitmap( const FileName : string)
8897:  Function ExtractIconCount( const FileName : string) : Integer
8898:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8899:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8900:  Procedure
     BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8901:  Procedure StretchTransfer(Dst:TJclBitmap32;
     DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
```

```
8902:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8903:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8904:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
       TGradientDirection) : Boolean;
8905:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
       RegionBitmapMode:TJclRegionBitmapMode): HRGN
8906:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8907:  Procedure ScreenShot1( bm : TBitmap);
8908:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8909:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8910:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8911:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8912:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8913:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8914:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8915:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8916:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8917:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8918:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8919:  Procedure Invert( Dst, Src : TJclBitmap32)
8920:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8921:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8922:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8923:  Procedure SetGamma( Gamma : Single)
8924: end;
8925:
8926: (*----------------------------------------------------------------------------*)
8927: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8928: begin
8929:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8930:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8931:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8932:  Function LockedDec( var Target : Integer) : Integer
8933:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8934:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8935:  Function LockedExchangeDec( var Target : Integer) : Integer
8936:  Function LockedExchangeInc( var Target : Integer) : Integer
8937:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8938:  Function LockedInc( var Target : Integer) : Integer
8939:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8940:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8941:   SIRegister_TJclDispatcherObject(CL);
8942:  Function WaitForMultipleObjects(const Objects:array of
       TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8943:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
       TimeOut : Cardinal):Cardinal
8944:   SIRegister_TJclCriticalSection(CL);
8945:   SIRegister_TJclCriticalSectionEx(CL);
8946:   SIRegister_TJclEvent(CL);
8947:   SIRegister_TJclWaitableTimer(CL);
8948:   SIRegister_TJclSemaphore(CL);
8949:   SIRegister_TJclMutex(CL);
8950:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8951:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8952:   SIRegister_TJclOptex(CL);
8953:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8954:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8955:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8956:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8957:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8958:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8959:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8960:   PMeteredSection', '^TMeteredSection // will not work
8961:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8962:   SIRegister_TJclMeteredSection(CL);
8963:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8964:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8965:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8966:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8967:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8968:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8969:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8970:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8971:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8972:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8973:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8974:   FindClass('TOBJECT'),'EJclCriticalSectionError
8975:   FindClass('TOBJECT'),'EJclEventError
8976:   FindClass('TOBJECT'),'EJclWaitableTimerError
8977:   FindClass('TOBJECT'),'EJclSemaphoreError
8978:   FindClass('TOBJECT'),'EJclMutexError
8979:   FindClass('TOBJECT'),'EJclMeteredSectionError
8980: end;
8981:
8982:
8983: //*************************unit uPSI_mORMotReport;
8984: Procedure SetCurrentPrinterAsDefault
8985: Function CurrentPrinterName : string
8986: Function mCurrentPrinterPaperSize : string
```

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

```
9076: { ------------------------------------------------------------------
9077:   Error handling
9078:   ------------------------------------------------------------------ }
9079: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9080: { Sets the error code }
9081: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9082: { Sets error code and default function value }
9083: function MathErr : Integer; external 'dmath';
9084: { Returns the error code }
9085: { ------------------------------------------------------------------
9086:   Dynamic arrays
9087:   ------------------------------------------------------------------ }
9088: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9089: { Sets the auto-initialization of arrays }
9090: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9091: { Creates floating point vector V[0..Ub] }
9092: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9093: { Creates integer vector V[0..Ub] }
9094: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9095: { Creates complex vector V[0..Ub] }
9096: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9097: { Creates boolean vector V[0..Ub] }
9098: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9099: { Creates string vector V[0..Ub] }
9100: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9101: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9102: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9103: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9104: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9105: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9106: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9107: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9108: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9109: { Creates string matrix A[0..Ub1, 0..Ub2] }
9110: { ------------------------------------------------------------------
9111:   Minimum, maximum, sign and exchange
9112:   ------------------------------------------------------------------ }
9113: function FMin(X, Y : Float) : Float; external 'dmath';
9114: { Minimum of 2 reals }
9115: function FMax(X, Y : Float) : Float; external 'dmath';
9116: { Maximum of 2 reals }
9117: function IMin(X, Y : Integer) : Integer; external 'dmath';
9118: { Minimum of 2 integers }
9119: function IMax(X, Y : Integer) : Integer; external 'dmath';
9120: { Maximum of 2 integers }
9121: function Sgn(X : Float) : Integer; external 'dmath';
9122: { Sign (returns 1 if X = 0) }
9123: function Sgn0(X : Float) : Integer; external 'dmath';
9124: { Sign (returns 0 if X = 0) }
9125: function DSgn(A, B : Float) : Float; external 'dmath';
9126: { Sgn(B) * |A| }
9127: procedure FSwap(var X, Y : Float); external 'dmath';
9128: { Exchange 2 reals }
9129: procedure ISwap(var X, Y : Integer); external 'dmath';
9130: { Exchange 2 integers }
9131: { ------------------------------------------------------------------
9132:   Rounding functions
9133:   ------------------------------------------------------------------ }
9134: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9135: { Rounds X to N decimal places }
9136: function Ceil(X : Float) : Integer; external 'dmath';
9137: { Ceiling function }
9138: function Floor(X : Float) : Integer; external 'dmath';
9139: { Floor function }
9140: { ------------------------------------------------------------------
9141:   Logarithms, exponentials and power
9142:   ------------------------------------------------------------------ }
9143: function Expo(X : Float) : Float; external 'dmath';
9144: { Exponential }
9145: function Exp2(X : Float) : Float; external 'dmath';
9146: { 2^X }
9147: function Exp10(X : Float) : Float; external 'dmath';
9148: { 10^X }
9149: function Log(X : Float) : Float; external 'dmath';
9150: { Natural log }
9151: function Log2(X : Float) : Float; external 'dmath';
9152: { Log, base 2 }
9153: function Log10(X : Float) : Float; external 'dmath';
9154: { Decimal log }
9155: function LogA(X, A : Float) : Float; external 'dmath';
9156: { Log, base A }
9157: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9158: { X^N }
9159: function Power(X, Y : Float) : Float; external 'dmath';
9160: { X^Y, X >= 0 }
9161: { ------------------------------------------------------------------
9162:   Trigonometric functions
9163:   ------------------------------------------------------------------ }
9164: function Pythag(X, Y : Float) : Float; external 'dmath';
```

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

```
9254: function DExpo(A, X : Float) : Float; external 'dmath';
9255: { Density of exponential distribution with parameter A }
9256: function FExpo(A, X : Float) : Float; external 'dmath';
9257: { Cumulative probability function for exponential dist. with parameter A }
9258: { ---------------------------------------------------------------
9259:   Standard normal distribution
9260:   --------------------------------------------------------------- }
9261: function DNorm(X : Float) : Float; external 'dmath';
9262: { Density of standard normal distribution }
9263: function FNorm(X : Float) : Float; external 'dmath';
9264: { Cumulative probability for standard normal distrib. }
9265: function PNorm(X : Float) : Float; external 'dmath';
9266: { Prob(|U| > X) for standard normal distrib. }
9267: function InvNorm(P : Float) : Float; external 'dmath';
9268: { Inverse of standard normal distribution }
9269: { ---------------------------------------------------------------
9270:   Student's distribution
9271:   --------------------------------------------------------------- }
9272: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9273: { Density of Student distribution with Nu d.o.f. }
9274: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9275: { Cumulative probability for Student distrib. with Nu d.o.f. }
9276: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9277: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9278: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9279: { Inverse of Student's t-distribution function }
9280: { ---------------------------------------------------------------
9281:   Khi-2 distribution
9282:   --------------------------------------------------------------- }
9283: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9284: { Density of Khi-2 distribution with Nu d.o.f. }
9285: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9286: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9287: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9288: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9289: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9290: { Inverse of Khi-2 distribution function }
9291: { ---------------------------------------------------------------
9292:   Fisher-Snedecor distribution
9293:   --------------------------------------------------------------- }
9294: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9295: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9296: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9297: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9298: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9299: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9300: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9301: { Inverse of Snedecor's F-distribution function }
9302: { ---------------------------------------------------------------
9303:   Beta distribution
9304:   --------------------------------------------------------------- }
9305: function DBeta(A, B, X : Float) : Float; external 'dmath';
9306: { Density of Beta distribution with parameters A and B }
9307: function FBeta(A, B, X : Float) : Float; external 'dmath';
9308: { Cumulative probability for Beta distrib. with param. A and B }
9309: { ---------------------------------------------------------------
9310:   Gamma distribution
9311:   --------------------------------------------------------------- }
9312: function DGamma(A, B, X : Float) : Float; external 'dmath';
9313: { Density of Gamma distribution with parameters A and B }
9314: function FGamma(A, B, X : Float) : Float; external 'dmath';
9315: { Cumulative probability for Gamma distrib. with param. A and B }
9316: { ---------------------------------------------------------------
9317:   Expression evaluation
9318:   --------------------------------------------------------------- }
9319: function InitEval : Integer; external 'dmath';
9320: { Initializes built-in functions and returns their number }
9321: function Eval(ExpressionString : String) : Float; external 'dmath';
9322: { Evaluates an expression at run-time }
9323: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9324: { Assigns a value to a variable }
9325: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9326: { Adds a function to the parser }
9327: { ---------------------------------------------------------------
9328:   Matrices and linear equations
9329:   --------------------------------------------------------------- }
9330: procedure GaussJordan(A           : TMatrix;
9331:                       Lb, Ub1, Ub2 : Integer;
9332:                       var Det      : Float); external 'dmath';
9333: { Transforms a matrix according to the Gauss-Jordan method }
9334: procedure LinEq(A       : TMatrix;
9335:                 B       : TVector;
9336:                 Lb, Ub  : Integer;
9337:                 var Det : Float); external 'dmath';
9338: { Solves a linear system according to the Gauss-Jordan method }
9339: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9340: { Cholesky factorization of a positive definite symmetric matrix }
9341: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9342: { LU decomposition }
```

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

```
9432: procedure Marquardt(Func      : TFuncNVar;
9433:                     HessGrad  : THessGrad;
9434:                     X         : TVector;
9435:                     Lb, Ub    : Integer;
9436:                     MaxIter   : Integer;
9437:                     Tol       : Float;
9438:                     var F_min : Float;
9439:                     G         : TVector;
9440:                     H_inv     : TMatrix;
9441:                     var Det   : Float); external 'dmath';
9442: { Minimization of a function of several variables (Marquardt's method) }
9443: procedure SaveMarquardt(FileName : string); external 'dmath';
9444: { Save Marquardt iterations in a file }
9445: procedure BFGS(Func      : TFuncNVar;
9446:                Gradient  : TGradient;
9447:                X         : TVector;
9448:                Lb, Ub    : Integer;
9449:                MaxIter   : Integer;
9450:                Tol       : Float;
9451:                var F_min : Float;
9452:                G         : TVector;
9453:                H_inv     : TMatrix); external 'dmath';
9454: { Minimization of a function of several variables (BFGS method) }
9455: procedure SaveBFGS(FileName : string); external 'dmath';
9456: { Save BFGS iterations in a file }
9457: procedure Simplex(Func      : TFuncNVar;
9458:                   X         : TVector;
9459:                   Lb, Ub    : Integer;
9460:                   MaxIter   : Integer;
9461:                   Tol       : Float;
9462:                   var F_min : Float); external 'dmath';
9463: { Minimization of a function of several variables (Simplex) }
9464: procedure SaveSimplex(FileName : string); external 'dmath';
9465: { Save Simplex iterations in a file }
9466: { ------------------------------------------------------------------
9467:   Nonlinear equations
9468:   ------------------------------------------------------------------ }
9469: procedure RootBrack(Func             : TFunc;
9470:                     var X, Y, FX, FY : Float); external 'dmath';
9471: { Brackets a root of function Func between X and Y }
9472: procedure Bisect(Func     : TFunc;
9473:                  var X, Y : Float;
9474:                  MaxIter  : Integer;
9475:                  Tol      : Float;
9476:                  var F    : Float); external 'dmath';
9477: { Bisection method }
9478: procedure Secant(Func     : TFunc;
9479:                  var X, Y : Float;
9480:                  MaxIter  : Integer;
9481:                  Tol      : Float;
9482:                  var F    : Float); external 'dmath';
9483: { Secant method }
9484: procedure NewtEq(Func, Deriv : TFunc;
9485:                  var X       : Float;
9486:                  MaxIter     : Integer;
9487:                  Tol         : Float;
9488:                  var F       : Float); external 'dmath';
9489: { Newton-Raphson method for a single nonlinear equation }
9490: procedure NewtEqs(Equations : TEquations;
9491:                   Jacobian  : TJacobian;
9492:                   X, F      : TVector;
9493:                   Lb, Ub    : Integer;
9494:                   MaxIter   : Integer;
9495:                   Tol       : Float); external 'dmath';
9496: { Newton-Raphson method for a system of nonlinear equations }
9497: procedure Broyden(Equations : TEquations;
9498:                   X, F      : TVector;
9499:                   Lb, Ub    : Integer;
9500:                   MaxIter   : Integer;
9501:                   Tol       : Float); external 'dmath';
9502: { Broyden's method for a system of nonlinear equations }
9503: { ------------------------------------------------------------------
9504:   Polynomials and rational fractions
9505:   ------------------------------------------------------------------ }
9506: function Poly(X    : Float;
9507:               Coef : TVector;
9508:               Deg  : Integer) : Float; external 'dmath';
9509: { Evaluates a polynomial }
9510: function RFrac(X          : Float;
9511:                Coef       : TVector;
9512:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9513: { Evaluates a rational fraction }
9514: function RootPol1(A, B  : Float;
9515:                   var X : Float) : Integer; external 'dmath';
9516: { Solves the linear equation A + B * X = 0 }
9517: function RootPol2(Coef : TVector;
9518:                   Z    : TCompVector) : Integer; external 'dmath';
9519: { Solves a quadratic equation }
9520: function RootPol3(Coef : TVector;
```

```
9521:                   Z    : TCompVector) : Integer; external 'dmath';
9522: { Solves a cubic equation }
9523: function RootPol4(Coef : TVector;
9524:                   Z    : TCompVector) : Integer; external 'dmath';
9525: { Solves a quartic equation }
9526: function RootPol(Coef : TVector;
9527:                  Deg  : Integer;
9528:                  Z    : TCompVector) : Integer; external 'dmath';
9529: { Solves a polynomial equation }
9530: function SetRealRoots(Deg : Integer;
9531:                        Z   : TCompVector;
9532:                        Tol : Float) : Integer; external 'dmath';
9533: { Set the imaginary part of a root to zero }
9534: procedure SortRoots(Deg : Integer;
9535:                     Z   : TCompVector); external 'dmath';
9536: { Sorts the roots of a polynomial }
9537: { ---------------------------------------------------------------
9538:   Numerical integration and differential equations
9539:   --------------------------------------------------------------- }
9540: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9541: { Integration by trapezoidal rule }
9542: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9543: { Integral from A to B }
9544: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9545: { Integral from 0 to B }
9546: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9547: { Convolution product at time T }
9548: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9549: { Convolution by trapezoidal rule }
9550: procedure RKF45(F                    : TDiffEqs;
9551:                 Neqn                 : Integer;
9552:                 Y, Yp                : TVector;
9553:                 var T                : Float;
9554:                 Tout, RelErr, AbsErr : Float;
9555:                 var Flag             : Integer); external 'dmath';
9556: { Integration of a system of differential equations }
9557: { ---------------------------------------------------------------
9558:   Fast Fourier Transform
9559:   --------------------------------------------------------------- }
9560: procedure FFT(NumSamples        : Integer;
9561:               InArray, OutArray : TCompVector); external 'dmath';
9562: { Fast Fourier Transform }
9563: procedure IFFT(NumSamples        : Integer;
9564:                InArray, OutArray : TCompVector); external 'dmath';
9565: { Inverse Fast Fourier Transform }
9566: procedure FFT_Integer(NumSamples     : Integer;
9567:                       RealIn, ImagIn : TIntVector;
9568:                       OutArray       : TCompVector); external 'dmath';
9569: { Fast Fourier Transform for integer data }
9570: procedure FFT_Integer_Cleanup; external 'dmath';
9571: { Clear memory after a call to FFT_Integer }
9572: procedure CalcFrequency(NumSamples,
9573:                         FrequencyIndex : Integer;
9574:                         InArray        : TCompVector;
9575:                         var FFT        : Complex); external 'dmath';
9576: { Direct computation of Fourier transform }
9577: { ---------------------------------------------------------------
9578:   Random numbers
9579:   --------------------------------------------------------------- }
9580: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9581: { Select generator }
9582: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9583: { Initialize generator }
9584: function IRanGen : RNG_IntType; external 'dmath';
9585: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9586: function IRanGen31 : RNG_IntType; external 'dmath';
9587: { 31-bit random integer in [0 .. 2^31 - 1] }
9588: function RanGen1 : Float; external 'dmath';
9589: { 32-bit random real in [0,1] }
9590: function RanGen2 : Float; external 'dmath';
9591: { 32-bit random real in [0,1) }
9592: function RanGen3 : Float; external 'dmath';
9593: { 32-bit random real in (0,1) }
9594: function RanGen53 : Float; external 'dmath';
9595: { 53-bit random real in [0,1) }
9596: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9597: { Initializes the 'Multiply with carry' random number generator }
9598: function IRanMWC : RNG_IntType; external 'dmath';
9599: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9600: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9601: { Initializes Mersenne Twister generator with a seed }
9602: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9603:                         KeyLength : Word); external 'dmath';
9604: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9605: function IRanMT : RNG_IntType; external 'dmath';
9606: { Random integer from MT generator }
9607: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9608: { Initializes the UVAG generator with a string }
9609: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
```

```
9610: { Initializes the UVAG generator with an integer }
9611: function IRanUVAG : RNG_IntType; external 'dmath';
9612: { Random integer from UVAG generator }
9613: function RanGaussStd : Float; external 'dmath';
9614: { Random number from standard normal distribution }
9615: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9616: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9617: procedure RanMult(M      : TVector; L      : TMatrix;
9618:                   Lb, Ub : Integer;
9619:                   X      : TVector); external 'dmath';
9620: { Random vector from multinormal distribution (correlated) }
9621: procedure RanMultIndep(M, S   : TVector;
9622:                        Lb, Ub : Integer;
9623:                        X      : TVector); external 'dmath';
9624: { Random vector from multinormal distribution (uncorrelated) }
9625: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9626: { Initializes Metropolis-Hastings parameters }
9627: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9628: { Returns Metropolis-Hastings parameters }
9629: procedure Hastings(Func      : TFuncNVar;
9630:                    T         : Float;
9631:                    X         : TVector;
9632:                    V         : TMatrix;
9633:                    Lb, Ub    : Integer;
9634:                    Xmat      : TMatrix;
9635:                    X_min     : TVector;
9636:                    var F_min : Float); external 'dmath';
9637: { Simulation of a probability density function by Metropolis-Hastings }
9638: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9639: { Initializes Simulated Annealing parameters }
9640: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9641: { Initializes log file }
9642: procedure SimAnn(Func          : TFuncNVar;
9643:                  X, Xmin, Xmax : TVector;
9644:                  Lb, Ub        : Integer;
9645:                  var F_min     : Float); external 'dmath';
9646: { Minimization of a function of several var. by simulated annealing }
9647: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9648: { Initializes Genetic Algorithm parameters }
9649: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9650: { Initializes log file }
9651: procedure GenAlg(Func          : TFuncNVar;
9652:                  X, Xmin, Xmax : TVector;
9653:                  Lb, Ub        : Integer;
9654:                  var F_min     : Float); external 'dmath';
9655: { Minimization of a function of several var. by genetic algorithm }
9656: { ------------------------------------------------------------------
9657:   Statistics
9658:   ------------------------------------------------------------------ }
9659: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9660: { Mean of sample X }
9661: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9662: { Minimum of sample X }
9663: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9664: { Maximum of sample X }
9665: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9666: { Median of sample X }
9667: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9668: { Standard deviation estimated from sample X }
9669: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9670: { Standard deviation of population }
9671: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9672: { Correlation coefficient }
9673: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9674: { Skewness of sample X }
9675: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9676: { Kurtosis of sample X }
9677: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9678: { Quick sort (ascending order) }
9679: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9680: { Quick sort (descending order) }
9681: procedure Interval(X1, X2             : Float;
9682:                    MinDiv, MaxDiv     : Integer;
9683:                    var Min, Max, Step : Float); external 'dmath';
9684: { Determines an interval for a set of values }
9685: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9686:                     var XMin, XMax, XStep : Float); external 'dmath';
9687: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9688: procedure StudIndep(N1, N2         : Integer;
9689:                     M1, M2, S1, S2 : Float;
9690:                     var T          : Float;
9691:                     var DoF        : Integer); external 'dmath';
9692: { Student t-test for independent samples }
9693: procedure StudPaired(X, Y    : TVector;
9694:                      Lb, Ub  : Integer;
9695:                      var T   : Float;
9696:                      var DoF : Integer); external 'dmath';
9697: { Student t-test for paired samples }
9698: procedure AnOVa1(Ns               : Integer;
```

```
9699:                 N               : TIntVector;
9700:                 M, S            : TVector;
9701:                 var V_f, V_r, F  : Float;
9702:                 var DoF_f, DoF_r : Integer); external 'dmath';
9703: { One-way analysis of variance }
9704: procedure AnOVa2(NA, NB, Nobs : Integer;
9705:                 M, S         : TMatrix;
9706:                 V, F         : TVector;
9707:                 DoF          : TIntVector); external 'dmath';
9708: { Two-way analysis of variance }
9709: procedure Snedecor(N1, N2         : Integer;
9710:                    S1, S2         : Float;
9711:                    var F          : Float;
9712:                    var DoF1, DoF2 : Integer); external 'dmath';
9713: { Snedecor's F-test (comparison of two variances) }
9714: procedure Bartlett(Ns       : Integer;
9715:                    N        : TIntVector;
9716:                    S        : TVector;
9717:                    var Khi2 : Float;
9718:                    var DoF  : Integer); external 'dmath';
9719: { Bartlett's test (comparison of several variances) }
9720: procedure Mann_Whitney(N1, N2     : Integer;
9721:                        X1, X2     : TVector;
9722:                        var U, Eps : Float); external 'dmath';
9723: { Mann-Whitney test}
9724: procedure Wilcoxon(X, Y       : TVector;
9725:                    Lb, Ub     : Integer;
9726:                    var Ndiff  : Integer;
9727:                    var T, Eps : Float); external 'dmath';
9728: { Wilcoxon test }
9729: procedure Kruskal_Wallis(Ns      : Integer;
9730:                          N       : TIntVector;
9731:                          X       : TMatrix;
9732:                          var H   : Float;
9733:                          var DoF : Integer); external 'dmath';
9734: { Kruskal-Wallis test }
9735: procedure Khi2_Conform(N_cls    : Integer;
9736:                        N_estim  : Integer;
9737:                        Obs      : TIntVector;
9738:                        Calc     : TVector;
9739:                        var Khi2 : Float;
9740:                        var DoF  : Integer); external 'dmath';
9741: { Khi-2 test for conformity }
9742: procedure Khi2_Indep(N_lin    : Integer;
9743:                      N_col    : Integer;
9744:                      Obs      : TIntMatrix;
9745:                      var Khi2 : Float;
9746:                      var DoF  : Integer); external 'dmath';
9747: { Khi-2 test for independence }
9748: procedure Woolf_Conform(N_cls   : Integer;
9749:                         N_estim : Integer;
9750:                         Obs     : TIntVector;
9751:                         Calc    : TVector;
9752:                         var G   : Float;
9753:                         var DoF : Integer); external 'dmath';
9754: { Woolf's test for conformity }
9755: procedure Woolf_Indep(N_lin   : Integer;
9756:                       N_col   : Integer;
9757:                       Obs     : TIntMatrix;
9758:                       var G   : Float;
9759:                       var DoF : Integer); external 'dmath';
9760: { Woolf's test for independence }
9761: procedure DimStatClassVector(var C : TStatClassVector;
9762:                              Ub    : Integer); external 'dmath';
9763: { Allocates an array of statistical classes: C[0..Ub] }
9764: procedure Distrib(X       : TVector;
9765:                   Lb, Ub  : Integer;
9766:                   A, B, H : Float;
9767:                   C       : TStatClassVector); external 'dmath';
9768: { Distributes an array X[Lb..Ub] into statistical classes }
9769: { ---------------------------------------------------------------
9770:   Linear / polynomial regression
9771:   ---------------------------------------------------------------- }
9772: procedure LinFit(X, Y   : TVector;
9773:                  Lb, Ub : Integer;
9774:                  B      : TVector;
9775:                  V      : TMatrix); external 'dmath';
9776: { Linear regression : Y = B(0) + B(1) * X }
9777: procedure WLinFit(X, Y, S : TVector;
9778:                   Lb, Ub  : Integer;
9779:                   B       : TVector;
9780:                   V       : TMatrix); external 'dmath';
9781: { Weighted linear regression : Y = B(0) + B(1) * X }
9782: procedure SVDLinFit(X, Y   : TVector;
9783:                     Lb, Ub : Integer;
9784:                     SVDTol : Float;
9785:                     B      : TVector;
9786:                     V      : TMatrix); external 'dmath';
9787: { Unweighted linear regression by singular value decomposition }
```

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

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

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

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

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

```
10233: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10234: { Plots a grid on the graph }
10235: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10236: { Writes the title of the graph }
10237: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10238: { Sets the maximum number of curves and re-initializes their parameters }
10239: procedure SetPointParam
10240: {$IFDEF DELPHI}
10241: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10242: {$ELSE}
10243: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10244: { Sets the point parameters for curve # CurvIndex }
10245: procedure SetLineParam
10246: {$IFDEF DELPHI}
10247: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10248: {$ELSE}
10249: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10250: { Sets the line parameters for curve # CurvIndex }
10251: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10252: { Sets the legend for curve # CurvIndex }
10253: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10254: { Sets the step for curve # CurvIndex }
10255: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10256: procedure GetPointParam
10257: {$IFDEF DELPHI}
10258: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10259: {$ELSE}
10260: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10261: { Returns the point parameters for curve # CurvIndex }
10262: procedure GetLineParam
10263: {$IFDEF DELPHI}
10264: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10265: {$ELSE}
10266: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10267: { Returns the line parameters for curve # CurvIndex }
10268: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10269: { Returns the legend for curve # CurvIndex }
10270: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10271: { Returns the step for curve # CurvIndex }
10272: {$IFDEF DELPHI}
10273: procedure PlotPoint(Canvas    : TCanvas;
10274:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10275: {$ELSE}
10276: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10277: {$ENDIF}
10278: { Plots a point on the screen }
10279: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10280:                     X, Y                  : TVector;
10281:                     Lb, Ub, CurvIndex     : Integer); external 'dmath';
10282: { Plots a curve }
10283: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10284:                                  X, Y, S             : TVector;
10285:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10286: { Plots a curve with error bars }
10287: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10288:                    Func                  : TFunc;
10289:                    Xmin, Xmax            : Float;
10290:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10291:                    CurvIndex             : Integer); external 'dmath';
10292: { Plots a function }
10293: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10294:                       NCurv               : Integer;
10295:                       ShowPoints, ShowLines : Boolean); external 'dmath';
10296: { Writes the legends for the plotted curves }
10297: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10298:                  Nx, Ny, Nc            : Integer;
10299:                  X, Y, Z               : TVector;
10300:                  F                     : TMatrix); external 'dmath';
10301: { Contour plot }
10302: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10303: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10304: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10305: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10306: {$IFNDEF DELPHI}
10307: procedure LeaveGraphics; external 'dmath';
10308: { Quits graphic mode }
10309: {$ENDIF}
10310: { ---------------------------------------------------------------
10311:   LaTeX graphics
10312:   --------------------------------------------------------------- }
10313: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10314:                           Header            : Boolean) : Boolean; external 'dmath';
10315: { Initializes the LaTeX file }
10316: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10317: { Sets the graphic window }
10318: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10319: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10320: { Sets the scale on the Ox axis }
10321: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
```

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

```
10411:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10412:   TTarModes', 'set of TTarMode
10413:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10414:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10415:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10416:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10417:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10418:   SIRegister_TTarArchive(CL);
10419:   SIRegister_TTarWriter(CL);
10420:  Function PermissionString( Permissions : TTarPermissions) : STRING
10421:  Function ConvertFilename( Filename : STRING) : STRING
10422:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10423:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10424:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10425: end;
10426:
10427:
10428: //************************************************unit uPSI_TlHelp32;
10429: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10430: begin
10431:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10432:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10433:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10434:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10435:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10436:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10437:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10438:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10439:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10440:   AddTypeS('THeapList32', 'tagHEAPLIST32
10441:  Const('HF32_DEFAULT','LongInt'( 1);
10442:  Const('HF32_SHARED','LongInt'( 2);
10443:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10444:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10445:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10446:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10447:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10448:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10449:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10450:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10451:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10452:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10453:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10454:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10455:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10456:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10457:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10458:    +'aPri : Longint; dwFlags : DWORD; end
10459:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10460:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10461:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10462:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10463: end;
10464:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10465:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10466:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10467:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10468:  Const('EWX_LOGOFF','LongInt'( 0);
10469:  Const('EWX_SHUTDOWN','LongInt'( 1);
10470:  Const('EWX_REBOOT','LongInt'( 2);
10471:  Const('EWX_FORCE','LongInt'( 4);
10472:  Const('EWX_POWEROFF','LongInt'( 8);
10473:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10474:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10475:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10476:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10477:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10478:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10479:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10480:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10481:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10482:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10483:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10484:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10485:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10486:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10487:  Function GetDesktopWindow : HWND
10488:  Function GetParent( hWnd : HWND) : HWND
10489:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10490:  Function GetTopWindow( hWnd : HWND) : HWND
10491:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10492:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10493:  //Delphi DFM
10494:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10495:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10496:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10497:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10498:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10499:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
```

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

```
10587:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10588:  (*Function FileExistsUTF8( const Filename : string) : boolean
10589:  Function FileAgeUTF8( const FileName : string) : Longint
10590:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10591:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10592:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10593:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10594:  Procedure FindCloseUTF8( var F : TSearchrec)
10595:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10596:  Function FileGetAttrUTF8( const FileName : String) : Longint
10597:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10598:  Function DeleteFileUTF8( const FileName : String) : Boolean
10599:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10600:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10601:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10602:  Function GetCurrentDirUTF8 : String
10603:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10604:  Function CreateDirUTF8( const NewDir : String) : Boolean
10605:  Function RemoveDirUTF8( const Dir : String) : Boolean
10606:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10607:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10608:  Function FileCreateUTF8( const FileName : string) : THandle;
10609:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10610:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10611:  Function FileSizeUtf8( const Filename : string) : int64
10612:  Function GetFileDescription( const AFilename : string) : string
10613:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10614:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10615:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10616:  Function IsUNCPath( const Path : String) : Boolean
10617:  Function ExtractUNCVolume( const Path : String) : String
10618:  Function ExtractFileRoot( FileName : String) : String
10619:  Function GetDarwinSystemFilename( Filename : string) : string
10620:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10621:  Function StrToCmdLineParam( const Param : string) : string
10622:  Function MergeCmdLineParams( ParamList : TStrings) : string
10623:  Procedure InvalidateFileStateCache( const Filename : string)
10624:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10625:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10626:  Function FindAllDocs(const Root, extmask: string): TStringlist;
10627:  Function ReadFileToString( const Filename : string) : string
10628:  procedure Inc1(var X: longint; N: Longint);
10629:
10630:  type
10631:   TCopyFileFlag = ( cffOverwriteFile,
10632:     cffCreateDestDirectory, cffPreserveTime );
10633:   TCopyFileFlags = set of TCopyFileFlag;*)
10634:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10635:   TCopyFileFlags', 'set of TCopyFileFlag
10636:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10637: end;
10638:
10639: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10640: begin
10641:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10642:   SIRegister_TMask(CL);
10643:   SIRegister_TParseStringList(CL);
10644:   SIRegister_TMaskList(CL);
10645:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10646:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10647:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10648:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10649: end;
10650:
10651: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10652: begin
10653:   //PShellHookInfo', '^TShellHookInfo // will not work
10654:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10655:   SHELLHOOKINFO', 'TShellHookInfo
10656:   LPSHELLHOOKINFO', 'PShellHookInfo
10657:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10658:   SIRegister_TJvShellHook(CL);
10659:  Function InitJvShellHooks : Boolean
10660:  Procedure UnInitJvShellHooks
10661: end;
10662:
10663: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10664: begin
10665:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10666:    +', dcHasSetSel, dcWantTab, dcNative )
10667:   TDlgCodes', 'set of TDlgCode
10668:  'dcWantMessage',' dcWantAllKeys);
10669:   SIRegister_IJvExControl(CL);
10670:   SIRegister_IJvDenySubClassing(CL);
10671:   SIRegister_TStructPtrMessage(CL);
10672:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10673:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10674:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10675:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
```

```
10676:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10677:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10678:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10679:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10680:  Function GetFocusedControl( AControl : TControl) : TWinControl
10681:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10682:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10683:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10684:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10685:   SIRegister_TJvExControl(CL);
10686:   SIRegister_TJvExWinControl(CL);
10687:   SIRegister_TJvExCustomControl(CL);
10688:   SIRegister_TJvExGraphicControl(CL);
10689:   SIRegister_TJvExHintWindow(CL);
10690:   SIRegister_TJvExPubGraphicControl(CL);
10691: end;
10692:
10693: (*----------------------------------------------------------------------------*)
10694: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10695: begin
10696:  Procedure EncodeStream( Input, Output : TStream)
10697:  Procedure DecodeStream( Input, Output : TStream)
10698:  Function EncodeString1( const Input : string) : string
10699:  Function DecodeString1( const Input : string) : string
10700: end;
10701:
10702: (*----------------------------------------------------------------------------*)
10703: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10704: begin
10705:   SIRegister_TWebAppRegInfo(CL);
10706:   SIRegister_TWebAppRegList(CL);
10707:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10708:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10709:  Procedure UnregisterWebApp( const AProgID : string)
10710:  Function FindRegisteredWebApp( const AProgID : string) : string
10711:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10712:  'sUDPPort','String 'UDPPort
10713: end;
10714:
10715: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10716: begin
10717:  // TStringDynArray', 'array of string
10718:  Function GetEnvVarValue( const VarName : string) : string
10719:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10720:  Function DeleteEnvVar( const VarName : string) : Integer
10721:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10722:  Function ExpandEnvVars( const Str : string) : string
10723:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10724:  Procedure GetAllEnvVarNames( const Names : TStrings);
10725:  Function GetAllEnvVarNames1 : TStringDynArray;
10726:  Function EnvBlockSize : Integer
10727:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10728:   SIRegister_TPJEnvVarsEnumerator(CL);
10729:   SIRegister_TPJEnvVars(CL);
10730:   FindClass('TOBJECT'),'EPJEnvVars
10731:   FindClass('TOBJECT'),'EPJEnvVars
10732:  //Procedure Register
10733: end;
10734:
10735: (*----------------------------------------------------------------------------*)
10736: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10737: begin
10738:  'cOneSecInMS','LongInt'( 1000);
10739:  //'cDefTimeSlice','LongInt'( 50);
10740:  //'cDefMaxExecTime',' cOneMinInMS);
10741:  'cAppErrorMask','LongInt'( 1 shl 29);
10742:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10743:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10744:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10745:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10746:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10747:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10748:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10749:   SIRegister_TPJCustomConsoleApp(CL);
10750:   SIRegister_TPJConsoleApp(CL);
10751: end;
10752:
10753: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10754: begin
10755:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10756:   t_encoding', '( uuencode, base64, mime )
10757:  Function internet_date( date : TDateTime) : string
10758:  Function lookup_hostname( const hostname : string) : longint
10759:  Function my_hostname : string
10760:  Function my_ip_address : longint
10761:  Function ip2string( ip_address : longint) : string
10762:  Function resolve_hostname( ip : longint) : string
10763:  Function address_from( const s : string; count : integer) : string
```

```
10764:  Function encode_base64( data : TStream) : TStringList
10765:  Function decode_base64( source : TStringList) : TMemoryStream
10766:  Function posn( const s, t : string; count : integer) : integer
10767:  Function poscn( c : char; const s : string; n : integer) : integer
10768:  Function filename_of( const s : string) : string
10769:  //Function trim( const s : string) : string
10770:  //Procedure setlength( var s : string; l : byte)
10771:  Function TimeZoneBias : longint
10772:  Function eight2seven_quoteprint( const s : string) : string
10773:  Function eight2seven_german( const s : string) : string
10774:  Function seven2eight_quoteprint( const s : string) : string end;
10775:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10776:  Function socketerror : cint
10777:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10778:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10779:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10780:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10781:  Function fplisten( s : cint; backlog : cint) : cint
10782:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10783:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10784:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10785:  Function NetAddrToStr( Entry : in_addr) : String
10786:  Function HostAddrToStr( Entry : in_addr) : String
10787:  Function StrToHostAddr( IP : String) : in_addr
10788:  Function StrToNetAddr( IP : String) : in_addr
10789:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10790:   cint8', 'shortint
10791:   cuint8', 'byte
10792:   cchar', 'cint8
10793:   cschar', 'cint8
10794:   cuchar', 'cuint8
10795:   cint16', 'smallint
10796:   cuint16', 'word
10797:   cshort', 'cint16
10798:   csshort', 'cint16
10799:   cushort', 'cuint16
10800:   cint32', 'longint
10801:   cuint32', 'longword
10802:   cint', 'cint32
10803:   csint', 'cint32
10804:   cuint', 'cuint32
10805:   csigned', 'cint
10806:   cunsigned', 'cuint
10807:   cint64', 'int64
10808:   clonglong', 'cint64
10809:   cslonglong', 'cint64
10810:   cbool', 'longbool
10811:   cfloat', 'single
10812:   cdouble', 'double
10813:   clongdouble', 'extended
10814:
10815: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10816: begin
10817:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10818:   SIRegister_TlkJSONdotnetclass(CL);
10819:   SIRegister_TlkJSONbase(CL);
10820:   SIRegister_TlkJSONnumber(CL);
10821:   SIRegister_TlkJSONstring(CL);
10822:   SIRegister_TlkJSONboolean(CL);
10823:   SIRegister_TlkJSONnull(CL);
10824:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10825:    +'se; data : TObject; var Continue : Boolean)
10826:   SIRegister_TlkJSONcustomlist(CL);
10827:   SIRegister_TlkJSONlist(CL);
10828:   SIRegister_TlkJSONobjectmethod(CL);
10829:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10830:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10831:   SIRegister_TlkHashTable(CL);
10832:   SIRegister_TlkBalTree(CL);
10833:   SIRegister_TlkJSONobject(CL);
10834:   SIRegister_TlkJSON(CL);
10835:   SIRegister_TlkJSONstreamed(CL);
10836:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10837: end;
10838:
10839: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10840: begin
10841:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10842:   SIRegister_TZSortedList(CL);
10843:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10844:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10845:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10846:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10847:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10848:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10849:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10850:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10851:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10852:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
```

```
10853:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10854:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10855:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10856:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10857:  Function StrToBoolEx( Str : string) : Boolean
10858:  Function BoolToStrEx( Bool : Boolean) : String
10859:  Function IsIpAddr( const Str : string) : Boolean
10860:  Function zSplitString( const Str, Delimiters : string) : TStrings
10861:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10862:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10863:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10864:  Function FloatToSQLStr( Value : Extended) : string
10865:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10866:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10867:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10868:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10869:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10870:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10871:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10872:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10873:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10874:  Function BytesToVar( const Value : TByteDynArray) : Variant
10875:  Function VarToBytes( const Value : Variant) : TByteDynArray
10876:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10877:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10878:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10879:  Function EncodeCString( const Value : string) : string
10880:  Function DecodeCString( const Value : string) : string
10881:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10882:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10883:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10884:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10885:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10886:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10887:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10888:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10889:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10890:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10891:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10892:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10893:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10894:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10895: end;
10896:
10897: unit uPSI_ZEncoding;
10898:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10899:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10900:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10901:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10902:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10903:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10904:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10905:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10906:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10907:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10908:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10909:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10910:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10911:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10912:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10913:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10914:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10915:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10916:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10917:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10918:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10919:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10920:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10921:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10922:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10923:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10924:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10925:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10926:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10927:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10928:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10929:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10930:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10931:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10932:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10933:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10934:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10935:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10936:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10937:  Function ZDefaultSystemCodePage : Word
10938:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10939:
```

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

```
11027:   PCharArr', 'array of PChar
11028:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
11029:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11030:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11031:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11032:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
11033:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
11034:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
11035:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11036:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11037:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11038:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11039:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11040: end;
11041:
11042: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11043: begin
11044:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11045:   SIRegister_TfrmBoldQueryUser(CL);
11046:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11047: end;
11048:
11049: (*----------------------------------------------------------------------------*)
11050: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11051: begin
11052:  //('befIsInDisplayList',' BoldElementFlag0);
11053:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11054:  //('befFollowerSelected',' BoldElementFlag2);
11055:   FindClass('TOBJECT'),'TBoldQueue
11056:   FindClass('TOBJECT'),'TBoldQueueable
11057:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11058:   SIRegister_TBoldQueueable(CL);
11059:   SIRegister_TBoldQueue(CL);
11060:  Function BoldQueueFinalized : Boolean
11061:  Function BoldInstalledQueue : TBoldQueue
11062: end;
11063:
11064: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11065: begin
11066:  const mmPerInch','Extended').setExtended( 25.4);
11067:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11068:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11069:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11070:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11071:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11072:   TBarLineType', '( white, black, black_half )
11073:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11074:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11075:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11076:   TCheckSumMethod', '( csmNone, csmModulo10 )
11077:   SIRegister_TAsBarcode(CL);
11078:  Function CheckSumModulo10( const data : string) : string
11079:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11080:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11081:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11082:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11083: end;
11084:
11085: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11086: begin
11087:    THomogeneousByteVector', 'array[0..3] of Byte
11088:   THomogeneousWordVector', 'array[0..3] of Word
11089:   THomogeneousIntVector', 'array[0..3] of Integer
11090:   THomogeneousFltVector', 'array[0..3] of single
11091:   THomogeneousDblVector', 'array[0..3] of double
11092:   THomogeneousExtVector', 'array[0..3] of extended
11093:   TAffineByteVector', 'array[0..2] of Byte
11094:   TAffineWordVector', 'array[0..2] of Word
11095:   TAffineIntVector', 'array[0..2] of Integer
11096:   TAffineFltVector', 'array[0..2] of single
11097:   TAffineDblVector', 'array[0..2] of double
11098:   TAffineExtVector', 'array[0..2] of extended
11099:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11100:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11101:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11102:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11103:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11104:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11105:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11106:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11107:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11108:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11109:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
```

```
11110:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11111:   TMatrix4b', 'THomogeneousByteMatrix
11112:   TMatrix4w', 'THomogeneousWordMatrix
11113:   TMatrix4i', 'THomogeneousIntMatrix
11114:   TMatrix4f', 'THomogeneousFltMatrix
11115:   TMatrix4d', 'THomogeneousDblMatrix
11116:   TMatrix4e', 'THomogeneousExtMatrix
11117:   TMatrix3b', 'TAffineByteMatrix
11118:   TMatrix3w', 'TAffineWordMatrix
11119:   TMatrix3i', 'TAffineIntMatrix
11120:   TMatrix3f', 'TAffineFltMatrix
11121:   TMatrix3d', 'TAffineDblMatrix
11122:   TMatrix3e', 'TAffineExtMatrix
11123:   //'PMatrix', '^TMatrix // will not work
11124:   TMatrixGL', 'THomogeneousFltMatrix
11125:   THomogeneousMatrix', 'THomogeneousFltMatrix
11126:   TAffineMatrix', 'TAffineFltMatrix
11127:   TQuaternion', 'record Vector : TVector4f; end
11128:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11129:   +'ger; Height : Integer; end
11130:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11131:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11132:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11133:  'EPSILON','Extended').setExtended( 1E-100);
11134:  'EPSILON2','Extended').setExtended( 1E-50);
11135:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11136:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11137:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11138:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11139:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11140:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11141:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11142:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11143:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11144:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11145:  Function VectorLength( V : array of Single) : Single
11146:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11147:  Procedure VectorNegate( V : array of Single)
11148:  Function VectorNorm( V : array of Single) : Single
11149:  Function VectorNormalize( V : array of Single) : Single
11150:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11151:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11152:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11153:  Procedure VectorScale( V : array of Single; Factor : Single)
11154:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11155:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11156:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11157:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11158:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11159:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11160:  Procedure MatrixAdjoint( var M : TMatrixGL)
11161:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11162:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11163:  Function MatrixDeterminant( M : TMatrixGL) : Single
11164:  Procedure MatrixInvert( var M : TMatrixGL)
11165:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11166:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11167:  Procedure MatrixTranspose( var M : TMatrixGL)
11168:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11169:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11170:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11171:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11172:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11173:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11174:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11175:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11176:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11177:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11178:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11179:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11180:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11181:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11182:  Function MakeAffineVector( V : array of Single) : TAffineVector
11183:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11184:  Function MakeVector( V : array of Single) : TVectorGL
11185:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11186:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11187:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11188:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11189:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11190:  Function ArcCosGL( X : Extended) : Extended
11191:  Function ArcSinGL( X : Extended) : Extended
11192:  Function ArcTan2GL( Y, X : Extended) : Extended
11193:  Function CoTanGL( X : Extended) : Extended
11194:  Function DegToRadGL( Degrees : Extended) : Extended
11195:  Function RadToDegGL( Radians : Extended) : Extended
11196:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11197:  Function TanGL( X : Extended) : Extended
11198:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
```

```
11199:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11200:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11201:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11202:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11203:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11204: end;
11205:
11206:
11207: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11208: begin
11209:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11210:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11211:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11212:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11213:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11214:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11215:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11216:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11217:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11218:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11219:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11220:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11221:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11222:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11223:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11224:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11225:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11226:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11227:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11228:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11229:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11230:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11231:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11232:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11233:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11234:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11235:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11236: end;
11237:
11238: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11239: begin
11240:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11241:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11242:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11243:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11244:   FindClass('TOBJECT'),'EInvalidParam
11245:  Function IsDCOMInstalled : Boolean
11246:  Function IsDCOMEnabled : Boolean
11247:  Function GetDCOMVersion : string
11248:  Function GetMDACVersion : string
11249:  Function GetMDACVersion2 : string
11250:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11251:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11252:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11253:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11254:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11255:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11256:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11257:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11258:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11259:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11260:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11261:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11262:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11263:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11264: end;
11265:
11266:
11267: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11268: begin
11269:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11270:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11271:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11272:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11273:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11274:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11275:  DegPerCycle','Extended').setExtended( 360.0);
11276:  DegPerGrad','Extended').setExtended( 0.9);
11277:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11278:  GradPerCycle','Extended').setExtended( 400.0);
11279:  GradPerDeg','Extended').setExtended( 1.1111111111111111111111111111111);
11280:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11281:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11282:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
```

```
11283:  RadPerGrad','Extended').setExtended( 0.01570796326794896619231321691639 8);
11284:  CyclePerDeg','Extended').setExtended( 0.00277777777777777777777777777777 778);
11285:  CyclePerGrad','Extended').setExtended( 0.0025);
11286:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11287:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11288:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11289:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11290:  Function MakePercentage( const Step, Max : Longint) : Longint
11291:  Function CelsiusToKelvin( const T : double) : double
11292:  Function CelsiusToFahrenheit( const T : double) : double
11293:  Function KelvinToCelsius( const T : double) : double
11294:  Function KelvinToFahrenheit( const T : double) : double
11295:  Function FahrenheitToCelsius( const T : double) : double
11296:  Function FahrenheitToKelvin( const T : double) : double
11297:  Function CycleToDeg( const Cycles : double) : double
11298:  Function CycleToGrad( const Cycles : double) : double
11299:  Function CycleToRad( const Cycles : double) : double
11300:  Function DegToCycle( const Degrees : double) : double
11301:  Function DegToGrad( const Degrees : double) : double
11302:  Function DegToRad( const Degrees : double) : double
11303:  Function GradToCycle( const Grads : double) : double
11304:  Function GradToDeg( const Grads : double) : double
11305:  Function GradToRad( const Grads : double) : double
11306:  Function RadToCycle( const Radians : double) : double
11307:  Function RadToDeg( const Radians : double) : double
11308:  Function RadToGrad( const Radians : double) : double
11309:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11310:  Function DmsToRad( const D, M : Integer; const S : double) : double
11311:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11312:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11313:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11314:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11315:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11316:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11317:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11318:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11319:  Function CmToInch( const Cm : double) : double
11320:  Function InchToCm( const Inch : double) : double
11321:  Function FeetToMetre( const Feet : double) : double
11322:  Function MetreToFeet( const Metre : double) : double
11323:  Function YardToMetre( const Yard : double) : double
11324:  Function MetreToYard( const Metre : double) : double
11325:  Function NmToKm( const Nm : double) : double
11326:  Function KmToNm( const Km : double) : double
11327:  Function KmToSm( const Km : double) : double
11328:  Function SmToKm( const Sm : double) : double
11329:  Function LitreToGalUs( const Litre : double) : double
11330:  Function GalUsToLitre( const GalUs : double) : double
11331:  Function GalUsToGalCan( const GalUs : double) : double
11332:  Function GalCanToGalUs( const GalCan : double) : double
11333:  Function GalUsToGalUk( const GalUs : double) : double
11334:  Function GalUkToGalUs( const GalUk : double) : double
11335:  Function LitreToGalCan( const Litre : double) : double
11336:  Function GalCanToLitre( const GalCan : double) : double
11337:  Function LitreToGalUk( const Litre : double) : double
11338:  Function GalUkToLitre( const GalUk : double) : double
11339:  Function KgToLb( const Kg : double) : double
11340:  Function LbToKg( const Lb : double) : double
11341:  Function KgToOz( const Kg : double) : double
11342:  Function OzToKg( const Oz : double) : double
11343:  Function CwtUsToKg( const Cwt : double) : double
11344:  Function CwtUkToKg( const Cwt : double) : double
11345:  Function KaratToKg( const Karat : double) : double
11346:  Function KgToCwtUs( const Kg : double) : double
11347:  Function KgToCwtUk( const Kg : double) : double
11348:  Function KgToKarat( const Kg : double) : double
11349:  Function KgToSton( const Kg : double) : double
11350:  Function KgToLton( const Kg : double) : double
11351:  Function StonToKg( const STon : double) : double
11352:  Function LtonToKg( const Lton : double) : double
11353:  Function QrUsToKg( const Qr : double) : double
11354:  Function QrUkToKg( const Qr : double) : double
11355:  Function KgToQrUs( const Kg : double) : double
11356:  Function KgToQrUk( const Kg : double) : double
11357:  Function PascalToBar( const Pa : double) : double
11358:  Function PascalToAt( const Pa : double) : double
11359:  Function PascalToTorr( const Pa : double) : double
11360:  Function BarToPascal( const Bar : double) : double
11361:  Function AtToPascal( const At : double) : double
11362:  Function TorrToPascal( const Torr : double) : double
11363:  Function KnotToMs( const Knot : double) : double
11364:  Function HpElectricToWatt( const HpE : double) : double
11365:  Function HpMetricToWatt( const HpM : double) : double
11366:  Function MsToKnot( const ms : double) : double
11367:  Function WattToHpElectric( const W : double) : double
11368:  Function WattToHpMetric( const W : double) : double
11369:  function getBigPI: string;    //PI of 1000 numbers
11370:
11371: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
```

```
11372: begin
11373:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11374:  Procedure CDCopyFile( const FileName, DestName : string)
11375:  Procedure CDMoveFile( const FileName, DestName : string)
11376:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11377:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11378:  Function CDGetTempDir : string
11379:  Function CDGetFileSize( FileName : string) : longint
11380:  Function GetFileTime( FileName : string) : longint
11381:  Function GetShortName( FileName : string) : string
11382:  Function GetFullName( FileName : string) : string
11383:  Function WinReboot : boolean
11384:  Function WinDir : String
11385:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11386:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11387:  Function devExecutor : TdevExecutor
11388: end;
11389:
11390: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11391: begin
11392:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11393:  Procedure Associate( Index : integer)
11394:  Procedure UnAssociate( Index : integer)
11395:  Function IsAssociated( Index : integer) : boolean
11396:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11397:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11398:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11399:  procedure RefreshIcons;
11400: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11401: function MergColor(Colors: Array of TColor): TColor;
11402: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11403: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11404: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11405: function GetInverseColor(AColor: TColor): TColor;
11406: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11407: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11408: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11409: Procedure GetSystemMenuFont(Font: TFont);
11410: end;
11411:
11412: //*************************unit uPSI_JvHLParser;*****************************
11413: function IsStringConstant(const St: string): Boolean;
11414: function IsIntConstant(const St: string): Boolean;
11415: function IsRealConstant(const St: string): Boolean;
11416: function IsIdentifier(const ID: string): Boolean;
11417: function GetStringValue(const St: string): string;
11418: procedure ParseString(const S: string; Ss: TStrings);
11419: function IsStringConstantW(const St: WideString): Boolean;
11420: function IsIntConstantW(const St: WideString): Boolean;
11421: function IsRealConstantW(const St: WideString): Boolean;
11422: function IsIdentifierW(const ID: WideString): Boolean;
11423: function GetStringValueW(const St: WideString): WideString;
11424: procedure ParseStringW(const S: WideString; Ss: TStrings);
11425:
11426:
11427: //*************************unit uPSI_JclMapi;*****************************
11428:
11429: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11430: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11431: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11432: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11433: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11434:
11435: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11436: begin
11437:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11438:  Function BuildType1Message( ADomain, AHost : String) : String
11439:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11440:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11441:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11442: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11443: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11444: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11445: end;
11446:
11447: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11448: begin
11449: ('IpAny','LongWord').SetUInt( $00000000);
11450:  IpLoopBack','LongWord').SetUInt( $7F000001);
11451:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11452:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11453:  PortAny','LongWord( $0000);
11454:  SocketMaxConnections','LongInt'( 5);
11455:  TIpAddr', 'LongWord
11456:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11457:  Function HostToNetLong( HostLong : LongWord) : LongWord
```

```
11458:  Function HostToNetShort( HostShort : Word) : Word
11459:  Function NetToHostLong( NetLong : LongWord) : LongWord
11460:  Function NetToHostShort( NetShort : Word) : Word
11461:  Function StrToIp( Ip : string) : TIpAddr
11462:  Function IpToStr( Ip : TIpAddr) : string
11463: end;
11464:
11465: (*----------------------------------------------------------------------------*)
11466: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11467: begin
11468:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11469:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11470:    +'amSha1, AlsmtpClientAuthAutoSelect )
11471:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11472:   SIRegister_TAlSmtpClient(CL);
11473: end;
11474:
11475: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11476: begin
11477: 'TBitNo', 'Integer
11478:   TStByteNo', 'Integer
11479: TStationNo', 'Integer
11480: TInOutNo', 'Integer
11481: TIo', '( EE, AA, NE, NA )
11482: TBitSet', 'set of TBitNo
11483: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11484: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11485: TBitAddr', 'LongInt
11486: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11487: TByteAddr', 'SmallInt
11488: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11489:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11490:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11491:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11492:  Function BitAddrToStr( Value : TBitAddr) : string
11493:  Function StrToBitAddr( const Value : string) : TBitAddr
11494:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11495:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11496:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11497:  Function ByteAddrToStr( Value : TByteAddr) : string
11498:  Function StrToByteAddr( const Value : string) : TByteAddr
11499:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11500:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11501:  Function InOutStateToStr( State : TInOutState) : string
11502:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11503:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11504: end;
11505:
11506: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11507: begin
11508: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11509:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11510: DpmiPmVector', 'Int64
11511:  'DInterval','LongInt'( 1000);
11512:  //'DEnabled','Boolean')BoolToStr( True);
11513:  'DIntFreq','string' if64
11514:  //'DMessages','Boolean if64);
11515:   SIRegister_TwdxCustomTimer(CL);
11516:   SIRegister_TwdxTimer(CL);
11517:   SIRegister_TwdxRtcTimer(CL);
11518:   SIRegister_TCustomIntTimer(CL);
11519:   SIRegister_TIntTimer(CL);
11520:   SIRegister_TRtcIntTimer(CL);
11521:  Function RealNow : TDateTime
11522:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11523:  Function DateTimeToMs( Time : TDateTime) : LongInt
11524: end;
11525:
11526: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11527: begin
11528: TIdSyslogPRI', 'Integer
11529: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11530:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11531:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11532:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11533:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11534: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11535:   SIRegister_TIdSysLogMsgPart(CL);
11536:   SIRegister_TIdSysLogMessage(CL);
11537:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11538:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11539:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11540:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11541:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11542:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11543: end;
11544:
11545: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
```

```
11546: begin
11547:  'UWhitespace','String '(?:\s*)
11548:  Function StripSpaces( const AText : string) : string
11549:  Function CharCount( const AText : string; Ch : Char) : Integer
11550:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11551:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11552: end;
11553:
11554:
11555: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11556: begin
11557:  ExtPascalVersion','String '0.9.8
11558:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11559:    +'Opera, brKonqueror, brMobileSafari )
11560:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11561:   AddTypeS('TExtProcedure', 'Procedure
11562:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11563:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11564:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11565:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11566:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11567:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11568:  Function StrToJS( const S : string; UseBR : boolean) : string
11569:  Function CaseOf( const S : string; const Cases : array of string) : integer
11570:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11571:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11572:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11573:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11574:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11575:  Function IsUpperCase( S : string) : boolean
11576:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11577:  Function BeautifyCSS( const AStyle : string) : string
11578:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11579:  Function JSDateToDateTime( JSDate : string) : TDateTime
11580: end;
11581:
11582: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11583: begin
11584:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11585:   TSHDeleteOptions', 'set of TSHDeleteOption
11586:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11587:   TSHRenameOptions', 'set of TSHRenameOption
11588:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11589:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11590:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11591:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11592:   TEnumFolderFlags', 'set of TEnumFolderFlag
11593:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11594:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11595:    +'IEnumIdList; Folder : IShellFolder; end
11596:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11597:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11598:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11599:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11600:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11601:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11602:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11603:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11604:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11605:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11606:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11607:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11608:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11609:  Function SHFreeMem( var P : Pointer) : Boolean
11610:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11611:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11612:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11613:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11614:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11615:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11616:  Function PidlFree( var IdList : PItemIdList) : Boolean
11617:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11618:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11619:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11620:  Function PidlToPath( IdList : PItemIdList) : string
11621:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11622:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11623:   PShellLink', '^TShellLink // will not work
11624:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11625:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11626:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11627:  Procedure ShellLinkFree( var Link : TShellLink)
11628:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11629:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11630:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11631:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11632:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11633:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11634:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
```

```
11635:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11636:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11637:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11638:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11639:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11640:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11641:  Function ShellOpenAs( const FileName : string) : Boolean
11642:  Function ShellRasDial( const EntryName : string) : Boolean
11643:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11644:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11645:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11646:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11647:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11648:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11649:  Function OemKeyScan( wOemChar : Word) : DWORD
11650:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11651: end;
11652:
11653: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11654: begin
11655:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11656:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11657:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11658:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11659:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11660:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11661:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11662:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11663:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11664:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11665:  Function xmlValidName( const Text : UnicodeString) : Boolean
11666:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11667:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11668:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11669:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11670:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11671:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11672:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11673:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11674:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11675:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11676:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11677:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11678:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11679:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11680:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11681:  Procedure SelfTestcXMLFunctions
11682: end;
11683:
11684: (*----------------------------------------------------------------------------*)
11685: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11686: begin
11687:  Function AWaitCursor : IUnknown
11688:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11689:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11690:  Function YesNo( const ACaption, AMsg : string) : boolean
11691:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11692:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11693:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11694:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11695:  Procedure GetSystemPaths( Strings : TStrings)
11696:  Procedure MakeEditNumeric( EditHandle : integer)
11697: end;
11698:
11699: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11700: begin
11701:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11702:  'BI_YUY2','LongWord( $32595559);
11703:  'BI_UYVY','LongWord').SetUInt( $59565955);
11704:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11705:  'BI_YVU9','LongWord').SetUInt( $39555659);
11706:  'BI_YUV12','LongWord( $30323449);
11707:  'BI_Y8','LongWord').SetUInt( $20203859);
11708:  'BI_Y211','LongWord').SetUInt( $31313259);
11709:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11710:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11711: end;
11712:
11713: (*----------------------------------------------------------------------------*)
11714: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11715: begin
11716:  'WM_USER','LongWord').SetUInt( $0400);
11717:  'WM_CAP_START','LongWord').SetUint($0400);
11718:  'WM_CAP_END','longword').SetUint($0400+85);
11719:  //WM_CAP_START+  85
11720:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11721:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11722:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
```

```
11723:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11724:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11725:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11726:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11727:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11728:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11729:  Function capGetUserData( hwnd : THandle) : LongInt
11730:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11731:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11732:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11733:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11734:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11735:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11736:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11737:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11738:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11739:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11740:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11741:  Function capEditCopy( hwnd : THandle) : LongInt
11742:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11743:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11744:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11745:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11746:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11747:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11748:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11749:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11750:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11751:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11752:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11753:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11754:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11755:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11756:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11757:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11758:  Function capGrabFrame( hwnd : THandle) : LongInt
11759:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11760:  Function capCaptureSequence( hwnd : THandle) : LongInt
11761:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11762:  Function capCaptureStop( hwnd : THandle) : LongInt
11763:  Function capCaptureAbort( hwnd : THandle) : LongInt
11764:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11765:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11766:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11767:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11768:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11769:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11770:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11771:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11772:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11773:  Function capPalettePaste( hwnd : THandle) : LongInt
11774:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11775:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11776:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11777:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11778:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11779:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11780:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11781:   //PCapStatus', '^TCapStatus // will not work
11782:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11783:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11784:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11785:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11786:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11787:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11788:    +' wNumAudioAllocated : WORD; end
11789:   //PCaptureParms', '^TCaptureParms // will not work
11790:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11791:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11792:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11793:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11794:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11795:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11796:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11797:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11798:    +'he : BOOL; AVStreamMaster : WORD; end
11799:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11800:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11801:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11802:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11803:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
       : Integer; hwndParent : THandle; nID : Integer) : THandle
11804:  Function
       capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11805:  'IDS_CAP_BEGIN','LongInt'( 300);
11806:  'IDS_CAP_END','LongInt'( 301);
11807:  'IDS_CAP_INFO','LongInt'( 401);
11808:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11809:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
```

```
11810:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11811:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11812:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11813:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11814:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11815:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11816:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11817:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11818:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11819:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11820:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11821:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11822:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11823:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11824:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11825:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11826:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11827:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11828:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11829:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11830:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11831:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11832:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11833:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11834:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11835:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11836:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11837:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11838:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11839:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11840:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11841:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11842:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11843:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11844:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11845:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11846:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11847:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11848:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11849:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11850:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11851:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11852:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11853:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11854:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11855:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11856:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11857:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11858:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11859:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11860:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11861:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11862:  'AVICAP32','String 'AVICAP32.dll
11863: end;
11864:
11865: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11866: begin
11867:  Function AlBoolToInt( Value : Boolean) : Integer
11868:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11869:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11870:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11871:  Function ALInc( var x : integer; Count : integer) : Integer
11872:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11873:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11874:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11875:  Function ALIsInteger(const S: AnsiString): Boolean;
11876:  function ALIsDecimal(const S: AnsiString): boolean;
11877:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11878:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11879:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11880:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11881:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11882:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11883:  Function ALRandomStr(const aLength: Longint): AnsiString;
11884:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11885:  Function ALRandomStrU(const aLength: Longint): String;
11886: end;
11887:
11888: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11889: begin
11890:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11891: end;
11892:
11893: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11894: begin
11895:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11896:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11897:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
```

```
11898:    +'; ullAvailExtendedVirtual : Int64; end
11899:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11900:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11901:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11902:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11903:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11904:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11905:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11906:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11907: end;
11908:
11909: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11910: begin
11911:   SIRegister_THandledObject(CL);
11912:   SIRegister_TEvent(CL);
11913:   SIRegister_TMutex(CL);
11914:   SIRegister_TSharedMem(CL);
11915:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11916:  'TRACE_BUFFER','String 'TRACE_BUFFER
11917:  'TRACE_MUTEX','String 'TRACE_MUTEX
11918:   //PTraceEntry', '^TTraceEntry // will not work
11919:   SIRegister_TIPCTracer(CL);
11920:  'MAX_CLIENTS','LongInt'( 6);
11921:  'IPCTIMEOUT','LongInt'( 2000);
11922:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11923:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11924:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11925:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11926:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11927:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11928:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11929:   FindClass('TOBJECT'),'EMonitorActive
11930:   FindClass('TOBJECT'),'TIPCThread
11931:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11932:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11933:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11934:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11935:   TClientFlags', 'set of TClientFlag
11936:   //PEventData', '^TEventData // will not work
11937:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11938:    +'lag; Flags : TClientFlags; end
11939:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11940:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11941:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11942:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11943:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11944:   SIRegister_TIPCEvent(CL);
11945:   //PClientDirRecords', '^TClientDirRecords // will not work
11946:   SIRegister_TClientDirectory(CL);
11947:   TIPCState', '( stInActive, stDisconnected, stConnected )
11948:   SIRegister_TIPCThread(CL);
11949:   SIRegister_TIPCMonitor(CL);
11950:   SIRegister_TIPCClient(CL);
11951:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11952: end;
11953:
11954: (*----------------------------------------------------------------------------*)
11955: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11956: begin
11957:   SIRegister_TAlGSMComm(CL);
11958:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11959:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11960:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11961:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11962:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11963:  end;
11964:
11965: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11966: begin
11967:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11968:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11969:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11970:   TInternetScheme', 'integer
11971:   TALIPv6Binary', 'array[1..16] of Char;
11972:  // TALIPv6Binary = array[1..16] of ansiChar;
11973:  //   TInternetScheme = Integer;
11974:   SIRegister_TALHTTPRequestHeader(CL);
11975:   SIRegister_TALHTTPCookie(CL);
11976:   SIRegister_TALHTTPCookieCollection(CL);
11977:   SIRegister_TALHTTPResponseHeader(CL);
11978:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11979:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11980: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
      Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11981: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
      Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11982: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
      Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
```

```
11983:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11984:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11985:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11986:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11987:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11988:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
       ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11989:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
       Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11990:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11991:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11992:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11993:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11994:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11995:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11996:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11997:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11998:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11999:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
12000:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
12001:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
12002:  Function ALZeroIpV6 : TALIPv6Binary
12003:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
12004:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
12005:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
12006:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
12007: end;
12008:
12009: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
12010: begin
12011:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
12012:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12013:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
12014:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12015:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
12016:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
12017:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
12018:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
12019:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
12020:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12021:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
12022: end;
12023:
12024: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12025: begin
12026:   SIRegister_TALEMailHeader(CL);
12027:   SIRegister_TALNewsArticleHeader(CL);
12028:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
12029:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
12030:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
12031:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
12032:  Function AlGenerateInternetMessageID : AnsiString;
12033:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
12034:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
12035:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12036: end;
12037:
12038: (*----------------------------------------------------------------------------*)
12039: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12040: begin
12041:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12042:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
12043:  Function ALgetLocalIPs : TALStrings
12044:  Function ALgetLocalHostName : AnsiString
12045: end;
12046:
12047: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12048: begin
12049:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
12050:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12051:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12052:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12053:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12054:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12055:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12056:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12057:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
```

```
12058: end;
12059:
12060: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12061: begin
12062:   TStartupInfoA', 'TStartupInfo
12063:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12064:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12065:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12066:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12067:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12068:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12069:  SE_TCB_NAME','String 'SeTcbPrivilege
12070:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12071:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12072:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12073:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12074:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12075:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12076:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12077:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12078:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12079:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12080:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12081:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12082:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12083:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12084:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12085:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12086:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12087:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12088:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12089:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12090:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12091:  Function AlGetEnvironmentString : AnsiString
12092:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12093:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12094:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12095:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12096:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12097: end;
12098:
12099: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12100: begin
12101:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12102:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12103:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12104:  Function ALGetModuleName : ansistring
12105:  Function ALGetModuleFileNameWithoutExtension : ansistring
12106:  Function ALGetModulePath : ansiString
12107:  Function AlGetFileSize( const AFileName : ansistring) : int64
12108:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12109:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12110:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12111:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12112:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12113:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12114:  Function ALFileExists( const Path : ansiString) : boolean
12115:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12116:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12117:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12118:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12119:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12120: end;
12121:
12122: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12123: begin
12124:   NativeInt', 'Integer
12125:   NativeUInt', 'Cardinal
12126:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12127:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12128:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12129:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12130:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12131:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12132:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12133:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12134:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12135:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12136:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12137:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
```

```
12138:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12139:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12140:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12141:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12142:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12143:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12144:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12145:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12146:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12147:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12148:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12149:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12150:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12151:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12152:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12153:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12154:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12155:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12156:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12157:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12158: end;
12159:
12160: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12161: begin
12162:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12163:   FindClass('TOBJECT'),'TALXMLNode
12164:   FindClass('TOBJECT'),'TALXMLNodeList
12165:   FindClass('TOBJECT'),'TALXMLDocument
12166:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12167:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12168:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12169:    +'nst Name : AnsiString; const Attributes : TALStrings)
12170:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12171:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12172:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12173:    +'ntDocType, ntDocFragment, ntNotation )
12174:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12175:   TALXMLDocOptions', 'set of TALXMLDocOption
12176:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12177:   TALXMLParseOptions', 'set of TALXMLParseOption
12178:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12179:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12180:   SIRegister_EALXMLDocError(CL);
12181:   SIRegister_TALXMLNodeList(CL);
12182:   SIRegister_TALXMLNode(CL);
12183:   SIRegister_TALXmlElementNode(CL);
12184:   SIRegister_TALXmlAttributeNode(CL);
12185:   SIRegister_TALXmlTextNode(CL);
12186:   SIRegister_TALXmlDocumentNode(CL);
12187:   SIRegister_TALXmlCommentNode(CL);
12188:   SIRegister_TALXmlProcessingInstrNode(CL);
12189:   SIRegister_TALXmlCDataNode(CL);
12190:   SIRegister_TALXmlEntityRefNode(CL);
12191:   SIRegister_TALXmlEntityNode(CL);
12192:   SIRegister_TALXmlDocTypeNode(CL);
12193:   SIRegister_TALXmlDocFragmentNode(CL);
12194:   SIRegister_TALXmlNotationNode(CL);
12195:   SIRegister_TALXMLDocument(CL);
12196:  cAlXMLUTF8EncodingStr','String 'UTF-8
12197:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12198:  CALNSDelim','String ':
12199:  CALXML','String 'xml
12200:  CALVersion','String 'version
12201:  CALEncoding','String 'encoding
12202:  CALStandalone','String 'standalone
12203:  CALDefaultNodeIndent','String '
12204:  CALXmlDocument','String 'DOCUMENT
12205:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12206:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
       EncodingStr:AnsiString);
12207:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
       ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12208:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
       ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12209:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12210:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12211:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12212: end;
12213:
```

```
12214: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12215: //based on TEEProc, TeCanvas, TEEngine, TChart
12216: begin
12217:  'TeePiStep','Double').setExtended( Pi / 180.0);
12218:  'TeeDefaultPerspective','LongInt'( 100);
12219:  'TeeMinAngle','LongInt'( 270);
12220:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12221:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12222:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12223:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12224:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12225:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12226:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12227:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12228:  'TA_LEFT','LongInt'( 0);
12229:  'TA_RIGHT','LongInt'( 2);
12230:  'TA_CENTER','LongInt'( 6);
12231:  'TA_TOP','LongInt'( 0);
12232:  'TA_BOTTOM','LongInt'( 8);
12233:  'teePATCOPY','LongInt'( 0);
12234:  'NumCirclePoints','LongInt'( 64);
12235:  'teeDEFAULT_CHARSET','LongInt'( 1);
12236:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12237:  'TA_LEFT','LongInt'( 0);
12238:  'bs_Solid','LongInt'( 0);
12239:  'teepf24Bit','LongInt'( 0);
12240:  'teepfDevice','LongInt'( 1);
12241:  'CM_MOUSELEAVE','LongInt'( 10000);
12242:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12243:  'DC_BRUSH','LongInt'( 18);
12244:  'DC_PEN','LongInt'( 19);
12245:   teeCOLORREF', 'LongWord
12246:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12247:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12248:   SIRegister_TFilterRegion(CL);
12249:   SIRegister_IFormCreator(CL);
12250:   SIRegister_TTeeFilter(CL);
12251:   //TFilterClass', 'class of TTeeFilter
12252:   SIRegister_TFilterItems(CL);
12253:   SIRegister_TConvolveFilter(CL);
12254:   SIRegister_TBlurFilter(CL);
12255:   SIRegister_TTeePicture(CL);
12256:   TPenEndStyle', '( esRound, esSquare, esFlat )
12257:   SIRegister_TChartPen(CL);
12258:   SIRegister_TChartHiddenPen(CL);
12259:   SIRegister_TDottedGrayPen(CL);
12260:   SIRegister_TDarkGrayPen(CL);
12261:   SIRegister_TWhitePen(CL);
12262:   SIRegister_TChartBrush(CL);
12263:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12264:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12265:   SIRegister_TView3DOptions(CL);
12266:   FindClass('TOBJECT'),'TTeeCanvas
12267:   TTeeTransparency', 'Integer
12268:   SIRegister_TTeeBlend(CL);
12269:   FindClass('TOBJECT'),'TCanvas3D
12270:   SIRegister_TTeeShadow(CL);
12271:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
       gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12272:   FindClass('TOBJECT'),'TSubGradient
12273:   SIRegister_TCustomTeeGradient(CL);
12274:   SIRegister_TSubGradient(CL);
12275:   SIRegister_TTeeGradient(CL);
12276:   SIRegister_TTeeFontGradient(CL);
12277:   SIRegister_TTeeFont(CL);
12278:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12279:   TCanvasTextAlign', 'Integer
12280:   TTeeCanvasHandle', 'HDC
12281:   SIRegister_TTeeCanvas(CL);
12282:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12283:   SIRegister_TFloatXYZ(CL);
12284:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12285:   TRGB', 'record blue: byte; green: byte; red: byte; end
12286:   {TRGB=packed record
12287:     Blue  : Byte;
12288:     Green : Byte;
12289:     Red   : Byte;
12290:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12291:
12292: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12293:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12294: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12295: TCanvas3DPlane', '( cpX, cpY, cpZ )
12296:   //IInterface', 'IUnknown
12297:   SIRegister_TCanvas3D(CL);
12298:   SIRegister_TTeeCanvas3D(CL);
12299:   TTrianglePoints', 'Array[0..2] of TPoint;
12300:   TFourPoints', 'Array[0..3] of TPoint;
12301:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
```

```
12302:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12303:  Function Point3D( const x, y, z : Integer) : TPoint3D
12304:  Procedure SwapDouble( var a, b : Double)
12305:  Procedure SwapInteger( var a, b : Integer)
12306:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12307:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12308:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12309:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12310:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12311:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12312:  Procedure UnClipCanvas( ACanvas : TCanvas)
12313:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12314:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12315:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12316:  'TeeCharForHeight','String 'W
12317:  'DarkerColorQuantity','Byte').SetUInt( 128);
12318:  'DarkColorQuantity','Byte').SetUInt( 64);
12319:   TButtonGetColorProc', 'Function  : TColor
12320:   SIRegister_TTeeButton(CL);
12321:   SIRegister_TButtonColor(CL);
12322:   SIRegister_TComboFlat(CL);
12323:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12324:  Function TeePoint( const aX, aY : Integer) : TPoint
12325:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12326:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12327:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12328:  Function OrientRectangle( const R : TRect) : TRect
12329:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12330:  Function PolygonBounds( const P : array of TPoint) : TRect
12331:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12332:  Function RGBValue( const Color : TColor) : TRGB
12333:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12334:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12335:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12336:  Function TeeCull( const P : TFourPoints) : Boolean;
12337:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12338:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12339:  Procedure SmoothStretch( Src, Dst : TBitmap);
12340:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12341:  Function TeeDistance( const x, y : Double) : Double
12342:  Function TeeLoadLibrary( const FileName : String) : HInst
12343:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12344:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12345:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12346:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
      Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12347:   SIRegister_ICanvasHyperlinks(CL);
12348:   SIRegister_ICanvasToolTips(CL);
12349:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12350: end;
12351:
12352: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12353: begin
12354:   TOvcHdc', 'Integer
12355:   TOvcHWND', 'Cardinal
12356:   TOvcHdc', 'HDC
12357:   TOvcHWND', 'HWND
12358:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12359:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12360:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12361:  Function DefaultEpoch : Integer
12362:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12363:  Procedure FixRealPrim( P : PChar; DC : Char)
12364:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12365:  Function GetLeftButton : Byte
12366:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12367:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12368:  Function GetShiftFlags : Byte
12369:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12370:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12371:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12372:  Function ovIsForegroundTask : Boolean
12373:  Function ovTrimLeft( const S : string) : string
12374:  Function ovTrimRight( const S : string) : string
12375:  Function ovQuotedStr( const S : string) : string
12376:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12377:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12378:  Function PtrDiff( const P1, P2 : PChar) : Word
12379:  Procedure PtrInc( var P, Delta : Word)
12380:  Procedure PtrDec( var P, Delta : Word)
12381:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12382:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
      SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12383:  Function ovMinI( X, Y : Integer) : Integer
12384:  Function ovMaxI( X, Y : Integer) : Integer
12385:  Function ovMinL( X, Y : LongInt) : LongInt
12386:  Function ovMaxL( X, Y : LongInt) : LongInt
12387:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12388:  Function PartialCompare( const S1, S2 : string) : Boolean
```

```
12389:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12390:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12391:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12392:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
        TransparentColor : TColor)
12393:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
        TransparentColor : TColorRef)
12394:  Function ovWidthOf( const R : TRect) : Integer
12395:  Function ovHeightOf( const R : TRect) : Integer
12396:  Procedure ovDebugOutput( const S : string)
12397:  Function GetArrowWidth( Width, Height : Integer) : Integer
12398:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12399:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12400:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12401:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12402:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12403:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12404:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12405:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12406:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12407:  Function CreateMetaFile( p1 : PChar) : HDC
12408:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12409:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12410:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12411:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12412:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12413:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12414:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12415:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12416:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12417:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12418:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12419:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12420:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
        SrcHeight:Int;Rop:DWORD):BOOL
12421:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12422:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
        SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12423:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12424:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12425:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12426:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12427:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12428:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12429:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12430:  Function UpdateColors( DC : HDC) : BOOL
12431:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12432:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12433:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12434:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12435:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12436:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12437:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
        HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12438:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
        yMask:Int):BOOL;
12439:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12440:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12441:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12442:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12443:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12444:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12445:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12446:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12447:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12448:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12449:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12450:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12451: end;
12452:
12453: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12454: begin
12455:   SIRegister_TOvcAbstractStore(CL);
12456:   //PExPropInfo', '^TExPropInfo // will not work
12457: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12458:   SIRegister_TOvcPropertyList(CL);
12459:   SIRegister_TOvcDataFiler(CL);
12460:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12461:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12462:  Function CreateStoredItem( const CompName, PropName : string) : string
12463:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12464:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12465: end;
12466:
12467: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12468: begin
12469:  'ovsetsize','LongInt'( 16);
12470:  'etSyntax','LongInt'( 0);
12471:  'etSymantic','LongInt'( 1);
```

```
12472:  'chCR','Char #13);
12473:  'chLF','Char #10);
12474:  'chLineSeparator',' chCR);
12475:   SIRegister_TCocoError(CL);
12476:   SIRegister_TCommentItem(CL);
12477:   SIRegister_TCommentList(CL);
12478:   SIRegister_TSymbolPosition(CL);
12479: TGenListType', '( glNever, glAlways, glOnError )
12480: TovBitSet', 'set of Integer
12481:   //PStartTable', '^TStartTable // will not work
12482: 'TovCharSet', 'set of AnsiChar
12483: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12484: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12485: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12486: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12487: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12488:    +'osition; const Data : string; ErrorType : integer)
12489: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12490: TGetCH', 'Function ( pos : longint) : char
12491: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12492:   SIRegister_TCocoRScanner(CL);
12493:   SIRegister_TCocoRGrammar(CL);
12494:  '_EF','Char #0);
12495:  '_TAB','Char').SetString( #09);
12496:  '_CR','Char').SetString( #13);
12497:  '_LF','Char').SetString( #10);
12498:  '_EL','').SetString( _CR);
12499:  '_EOF','Char').SetString( #26);
12500:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12501:  'minErrDist','LongInt'( 2);
12502:  Function ovPadL( S : string; ch : char; L : integer) : string
12503: end;
12504:
12505:   TFormatSettings = record
12506:     CurrencyFormat: Byte;
12507:     NegCurrFormat: Byte;
12508:     ThousandSeparator: Char;
12509:     DecimalSeparator: Char;
12510:     CurrencyDecimals: Byte;
12511:     DateSeparator: Char;
12512:     TimeSeparator: Char;
12513:     ListSeparator: Char;
12514:     CurrencyString: string;
12515:     ShortDateFormat: string;
12516:     LongDateFormat: string;
12517:     TimeAMString: string;
12518:     TimePMString: string;
12519:     ShortTimeFormat: string;
12520:     LongTimeFormat: string;
12521:     ShortMonthNames: array[1..12] of string;
12522:     LongMonthNames: array[1..12] of string;
12523:     ShortDayNames: array[1..7] of string;
12524:     LongDayNames: array[1..7] of string;
12525:     TwoDigitYearCenturyWindow: Word;
12526:    end;
12527:
12528: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12529: begin
12530:  Function ovFormatSettings : TFormatSettings
12531: end;
12532:
12533: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12534: begin
12535:   TOvcCharSet', 'set of Char
12536:   ovBTable', 'array[0..255] of Byte
12537:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
      HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12538:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12539:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12540:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12541:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12542:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12543:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12544:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12545:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12546:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12547:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12548:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12549:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12550:  Function LoCaseChar( C : Char) : Char
12551:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12552:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12553:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12554:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12555:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12556:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12557:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12558:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12559:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
```

```
12560:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12561:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12562:  Procedure TrimAllSpacesPChar( P : PChar)
12563:  Function TrimEmbeddedZeros( const S : string) : string
12564:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12565:  Function TrimTrailPrimPChar( S : PChar) : PChar
12566:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12567:  Function TrimTrailingZeros( const S : string) : string
12568:  Procedure TrimTrailingZerosPChar( P : PChar)
12569:  Function UpCaseChar( C : Char) : Char
12570:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12571:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12572:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12573: end;
12574:
12575: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12576: begin
12577:   //PRaiseFrame', '^TRaiseFrame // will not work
12578:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12579:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12580:  Procedure SafeCloseHandle( var Handle : THandle)
12581:  Procedure ExchangeInteger( X1, X2 : Integer)
12582:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12583:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12584:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12585:
12586: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12587:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12588: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12589:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12590:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12591:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12592:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12593:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12594:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12595:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12596:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12597:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12598:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12599:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12600:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12601:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12602:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12603:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12604:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12605:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12606:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12607:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12608:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12609:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12610:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12611:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12612:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12613:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12614:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12615:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12616:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12617:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12618:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12619:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12620:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12621:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12622:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12623:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12624:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12625:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12626:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12627:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12628:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12629:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12630:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12631:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12632:       var lpLuid: TLargeInteger): BOOL; stdcall;
12633:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12634:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12635:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12636:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12637:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12638:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12639:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12640:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12641:       var GenerateOnClose: BOOL): BOOL; stdcall;
12642:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12643:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12644:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12645:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12646:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12647:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12648:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
```

```
12649:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12650:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12651:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12652:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12653:       var phkResult: HKEY): Longint; stdcall;
12654:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12655:       var phkResult: HKEY): Longint; stdcall;
12656:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12657:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12658:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12659:       lpdwDisposition: PDWORD): Longint; stdcall;
12660:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12661:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12662:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12663:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12664:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12665:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12666:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12667:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12668:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12669:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12670:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12671:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12672:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12673:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12674:       lpcbClass: PDWORD; lpReserved: Pointer;
12675:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12676:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12677:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12678:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12679:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12680:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12681:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12682:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12683:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12684:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12685:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12686:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12687:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12688:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12689:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12690:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12691:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12692:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12693:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12694:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12695:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12696:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12697:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12698:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12699:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12700:
12701:  Function wAddAtom( lpString : PKOLChar) : ATOM
12702:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12703:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
       lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12704:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12705:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
       lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12706:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12707:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
       TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12708:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12709:  Function wCreateDirectoryEx(lpTemplateDirectory,
       lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12710:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
       bInitialState:BOOL;lpName:PKOLChar):THandle;
12711:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
       PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12712:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
       dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12713:  Function wCreateHardLink(lpFileName,
       lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12714:  Function
       CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12715:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
       nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12716:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
       lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
       Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
       lpProcessInfo:TProcessInformation):BOOL
12717:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
       Longint; lpName : PKOLChar) : THandle
12718:  Function
       wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12719:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12720:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12721:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12722:  //Function
       wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
```

```
12723:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12724:  //Function
        wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12725:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12726:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12727:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12728:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12729:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12730:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12731:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
        dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12732:  Function wFindAtom( lpString : PKOLChar) : ATOM
12733:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12734:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12735:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12736:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12737:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12738:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12739:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12740:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12741:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12742:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12743:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12744:  Function wGetCommandLine : PKOLChar
12745:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12746:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12747:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12748:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12749:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12750:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12751:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12752:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12753:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12754:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12755:  Function wGetEnvironmentStrings : PKOLChar
12756:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12757:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12758:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12759:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12760:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12761:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12762:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12763:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12764:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12765:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12766:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12767:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12768:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12769:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12770:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12771:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12772:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12773:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12774:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12775: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12776:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12777:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12778:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12779: //Function
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12780: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12781: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
        : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
        lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12782:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12783:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12784:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12785:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12786:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12787:  Function
        wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12788:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12789:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
```

```
12790:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12791:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12792:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
        TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12793:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12794:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12795:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12796:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12797:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12798:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12799:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsRead:DWORD):BOOL;
12800:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12801:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12802:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
        lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12803:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
        lpNumbOfEventsRead:DWORD):BOOL;
12804:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
        : TCoord; var lpReadRegion : TSmallRect) : BOOL
12805:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12806:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12807:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
        lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12808:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12809:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12810:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12811:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12812:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12813:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12814:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12815:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12816:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12817:  //Function wUpdateResource(hUpdate:THandle;lpType,
        lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12818:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12819:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12820:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
        DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12821:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
        var lpNumberOfEventsWritten : DWORD) : BOOL
12822:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
        TCoord; var lpWriteRegion : TSmallRect) : BOOL
12823:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12824:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12825:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12826:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12827:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12828:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12829:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12830:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12831:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12832:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12833:  Function wlstrlen( lpString : PKOLChar) : Integer
12834:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
        PNetConnectInfoStruct) : DWORD
12835:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12836:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12837:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12838:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12839:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12840:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12841:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12842:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12843:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12844:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12845:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12846:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12847:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12848: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12849:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12850: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12851: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12852:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12853:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12854:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12855:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12856:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
```

```
12857: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12858: //Func wGetPrivateProfileStruct(lpszSection,
       lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12859: //Func wWritePrivateProfileStruct(lpszSection,
       lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12860:  Function wAddFontResource( FileName : PKOLChar) : Integer
12861:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12862:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12863:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12864:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12865:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12866: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12867:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
       fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
       fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12868:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12869:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12870: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12871:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12872:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12873: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
       pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12874: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12875:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12876:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12877:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12878:  //Function wExtTextOut(DC:HDC;X,
       Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12879:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12880:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12881:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12882:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12883: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12884: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12885:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12886:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12887: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12888: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
       lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12889:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12890: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12891:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12892: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12893:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12894:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12895:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12896:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12897:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12898:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12899:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12900:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12901:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12902:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12903:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12904:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12905:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12906:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12907:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12908:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12909:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12910:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12911:  //Function
       wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12912:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12913: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
       dwFlags : DWORD; lParam : Pointer) : Longint
12914:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12915:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12916:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12917:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12918:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12919:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12920: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12921:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12922:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12923:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12924:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12925:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12926:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12927:  //Function wCreateDesktop(lpszDesktop,
       lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12928:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
       HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12929:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
       lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12930:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
       hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12931: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
       nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
```

```
12932:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
        dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12933:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12934:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12935:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12936:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12937:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
        : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12938:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12939:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12940:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
12941:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
        nIDStaticPath:Int;uFiletype:UINT):Int;
12942:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12943:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12944: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
        cy:Int;Flags:UINT):BOOL;
12945:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12946:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12947:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12948:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
        pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12949: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12950:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12951:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12952:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12953:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12954:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12955:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12956:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12957:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12958:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12959:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12960:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12961:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
        lpnTabStopPositions) : DWORD
12962: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
        lpnLengthNeed:DWORD)BOOL;
12963:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12964:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12965:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12966:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12967: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
        nHeigt:Int):BOOL;
12968:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12969:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12970:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12971:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12972:  Function wIsCharLower( ch : KOLChar) : BOOL
12973:  Function wIsCharUpper( ch : KOLChar) : BOOL
12974:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12975:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12976:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12977:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12978:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12979:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12980:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12981:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12982:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12983:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12984:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12985:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12986:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12987:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12988:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12989:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12990:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12991:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
12992:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
12993:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12994:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12995:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12996:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
12997:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
12998:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12999:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13000:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
13001: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
13002: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
13003:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
13004: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13005:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
13006:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
13007:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13008:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
13009:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
        lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
```

```
13010:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
       lpdwResult:DWORD): LRESULT
13011:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13012:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
13013:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
13014:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13015:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13016: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13017:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
13018:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13019:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
13020:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13021: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
13022:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
       lpnTabStopPositions,nTabOrigin:Int):Longint;
13023:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
13024:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13025:  Function wVkKeyScan( ch : KOLChar) : SHORT
13026:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13027:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
13028:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
13029:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
13030:
13031: //TestDrive!
13032: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13033:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13034:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
13035:  Function GetLocalUserSidStr( const UserName : string) : string
13036:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
13037:  Function Impersonate2User( const domain : string; const user : string) : boolean
13038:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
13039:  Function KillProcessbyname( const exename : string; var found : integer) : integer
13040:  Function getWinProcessList : TStringList
13041:  function WaitTilClose(hWnd: Integer): Integer;
13042:  function DoUserMsgs: Boolean;
13043:  function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13044:  procedure ShowMsg(hParent: Integer; const Mess, Title: String);  //modal but NOT blockable
13045:  procedure DeleteMsgForm(Handle: Integer);
13046:  procedure DisableForms;
13047:  function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13048: end;
13049:
13050: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13051: begin
13052:  'AfMaxSyncSlots','LongInt'( 64);
13053:  'AfSynchronizeTimeout','LongInt'( 2000);
13054:   TAfSyncSlotID', 'DWORD
13055:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13056:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13057:   TAfSafeDirectSyncEvent', 'Procedure
13058:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13059:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13060:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13061:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13062:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13063:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13064:  Function AfIsSyncMethod : Boolean
13065:  Function AfSyncWnd : HWnd
13066:  Function AfSyncStatistics : TAfSyncStatistics
13067:  Procedure AfClearSyncStatistics
13068: end;
13069:
13070: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13071: begin
13072:  'fBinary','LongWord')( $00000001);
13073:  'fParity','LongWord'( $00000002);
13074:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13075:  'fOutxDsrFlow','LongWord')( $00000008);
13076:  'fDtrControl','LongWord')( $00000030);
13077:  'fDtrControlDisable','LongWord')( $00000000);
13078:  'fDtrControlEnable','LongWord')( $00000010);
13079:  'fDtrControlHandshake','LongWord')( $00000020);
13080:  'fDsrSensitivity','LongWord')( $00000040);
13081:  'fTXContinueOnXoff','LongWord')( $00000080);
13082:  'fOutX','LongWord')( $00000100);
13083:  'fInX','LongWord')( $00000200);
13084:  'fErrorChar','LongWord')( $00000400);
13085:  'fNull','LongWord')( $00000800);
13086:  'fRtsControl','LongWord')( $00003000);
13087:  'fRtsControlDisable','LongWord')( $00000000);
13088:  'fRtsControlEnable','LongWord')( $00001000);
13089:  'fRtsControlHandshake','LongWord')( $00002000);
13090:  'fRtsControlToggle','LongWord')( $00003000);
13091:  'fAbortOnError','LongWord')( $00004000);
13092:  'fDummy2','LongWord')( $FFFF8000);
13093:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13094:   FindClass('TOBJECT'),'EAfComPortCoreError
13095:   FindClass('TOBJECT'),'TAfComPortCore
13096:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
```

```
13097:    +'tKind : TAfCoreEvent; Data : DWORD)
13098:   SIRegister_TAfComPortCoreThread(CL);
13099:   SIRegister_TAfComPortEventThread(CL);
13100:   SIRegister_TAfComPortWriteThread(CL);
13101:   SIRegister_TAfComPortCore(CL);
13102:  Function FormatDeviceName( PortNumber : Integer) : string
13103: end;
13104:
13105: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13106: begin
13107:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13108:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13109:   SIRegister_TApplicationFileIO(CL);
13110:   TDataFileCapability', '( dfcRead, dfcWrite )
13111:   TDataFileCapabilities', 'set of TDataFileCapability
13112:   SIRegister_TDataFile(CL);
13113:   //TDataFileClass', 'class of TDataFile
13114:  Function ApplicationFileIODefined : Boolean
13115:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13116:  Function FileStreamExists(const fileName: String) : Boolean
13117:  //Procedure Register
13118: end;
13119:
13120: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13121: begin
13122:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13123:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13124:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13125:   TALFBXScale', 'Integer
13126:   FindClass('TOBJECT'),'EALFBXConvertError
13127:   SIRegister_EALFBXError(CL);
13128:   SIRegister_EALFBXException(CL);
13129:   FindClass('TOBJECT'),'EALFBXGFixError
13130:   FindClass('TOBJECT'),'EALFBXDSQLError
13131:   FindClass('TOBJECT'),'EALFBXDynError
13132:   FindClass('TOBJECT'),'EALFBXGBakError
13133:   FindClass('TOBJECT'),'EALFBXGSecError
13134:   FindClass('TOBJECT'),'EALFBXLicenseError
13135:   FindClass('TOBJECT'),'EALFBXGStatError
13136:   //EALFBXExceptionClass', 'class of EALFBXError
13137:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13138:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13139:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13140:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13141:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13142:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13143:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13144:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13145:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13146:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13147:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13148:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13149:   TALFBXTransParams', 'set of TALFBXTransParam
13150:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13151:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13152:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13153:  'cALFBXMaxParamLength','LongInt'( 125);
13154:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13155:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13156:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13157:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13158:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13159:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13160:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13161:   SIRegister_TALFBXSQLDA(CL);
13162:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13163:   SIRegister_TALFBXPoolStream(CL);
13164:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13165:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13166:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13167:   //TALFBXArrayDesc', 'TISCArrayDesc
13168:   //TALFBXBlobDesc', 'TISCBlobDesc
13169:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13170:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13171:   SIRegister_TALFBXSQLResult(CL);
13172:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13173:   SIRegister_TALFBXSQLParams(CL);
13174:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13175:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13176:    +'atementType : TALFBXStatementType; end
13177:   FindClass('TOBJECT'),'TALFBXLibrary
13178:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13179:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13180:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13181:    //+' Excep : EALFBXExceptionClass)
13182:   SIRegister_TALFBXLibrary(CL);
13183:  'cAlFBXDateOffset','LongInt'( 15018);
13184:  'cALFBXTimeCoeff','LongInt'( 864000000);
13185:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
```

```
13186:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13187:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13188:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13189:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13190:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13191:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13192:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13193:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13194:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13195:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13196:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13197:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13198:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13199: end;
13200:
13201: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13202: begin
13203:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13204:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13205:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13206:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13207:    +'teger; First : Integer; CacheThreshold : Integer; end
13208:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13209:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13210:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13211:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13212:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13213:   SIRegister_TALFBXClient(CL);
13214:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13215:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13216:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13217:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13218:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13219:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13220:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13221:   SIRegister_TALFBXConnectionPoolClient(CL);
13222:   SIRegister_TALFBXEventThread(CL);
13223:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13224: end;
13225:
13226: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13227: begin
13228: _OSVERSIONINFOA = record
13229:     dwOSVersionInfoSize: DWORD;
13230:     dwMajorVersion: DWORD;
13231:     dwMinorVersion: DWORD;
13232:     dwBuildNumber: DWORD;
13233:     dwPlatformId: DWORD;
13234:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13235:   end;
13236:  TOSVersionInfoA', '_OSVERSIONINFOA
13237:   TOSVersionInfo', 'TOSVersionInfoA
13238:  'WS_EX_RIGHT','LongWord')( $00001000);
13239:  'WS_EX_LEFT','LongWord')( $00000000);
13240:  'WS_EX_RTLREADING','LongWord')( $00002000);
13241:  'WS_EX_LTRREADING','LongWord')( $00000000);
13242:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13243:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13244:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13245:  'LAYOUT_RTL','LongWord')( $00000001);
13246:  'LAYOUT_BTT','LongWord')( $00000002);
13247:  'LAYOUT_VBH','LongWord')( $00000004);
13248:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13249:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13250:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13251:  Function GetLayout( dc : hdc) : DWORD
13252:  Function IsBidi : Boolean
13253:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13254:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13255:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13256:  Function GetPriorityClass( hProcess : THandle) : DWORD
13257:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13258:  Function CloseClipboard : BOOL
13259:  Function GetClipboardSequenceNumber : DWORD
13260:  Function GetClipboardOwner : HWND
13261:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13262:  Function GetClipboardViewer : HWND
13263:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13264:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13265:  Function GetClipboardData( uFormat : UINT) : THandle
13266:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13267:  Function CountClipboardFormats : Integer
13268:  Function EnumClipboardFormats( format : UINT) : UINT
13269:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13270:  Function EmptyClipboard : BOOL
13271:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13272:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13273:  Function GetOpenClipboardWindow : HWND
13274:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
```

```
13275:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13276:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13277:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13278:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13279:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13280:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13281:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13282:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13283: end;
13284:
13285: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13286: begin
13287:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13288:  Function GetTemporaryFilesPath : String
13289:  Function GetTemporaryFileName : String
13290:  Function FindFileInPaths( const fileName, paths : String) : String
13291:  Function PathsToString( const paths : TStrings) : String
13292:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13293:  //Function MacroExpandPath( const aPath : String) : String
13294: end;
13295:
13296: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13297: begin
13298:   SIRegister_TALMultiPartBaseContent(CL);
13299:   SIRegister_TALMultiPartBaseContents(CL);
13300:   SIRegister_TAlMultiPartBaseStream(CL);
13301:   SIRegister_TALMultipartBaseEncoder(CL);
13302:   SIRegister_TALMultipartBaseDecoder(CL);
13303:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13304:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13305:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13306: end;
13307:
13308: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13309: begin
13310:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13311:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13312:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13313:  Function aAllocPadedMem( Size : Cardinal) : TObject
13314:  Procedure aFreePadedMem( var P : TObject);
13315:  Procedure aFreePadedMem1( var P : PChar);
13316:  Function aCheckPadedMem( P : Pointer) : Byte
13317:  Function aGetPadMemSize( P : Pointer) : Cardinal
13318:  Function aAllocMem( Size : Cardinal) : Pointer
13319:  Function aStrLen( const Str : PChar) : Cardinal
13320:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13321:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13322:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13323:  Function aStrEnd( const Str : PChar) : PChar
13324:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13325:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13326:  Function aPCharLength( const Str : PChar) : Cardinal
13327:  Function aPCharUpper( Str : PChar) : PChar
13328:  Function aPCharLower( Str : PChar) : PChar
13329:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13330:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13331:  Function aCopyTail( const S : String; Len : Integer) : String
13332:  Function aInt2Thos( I : Int64) : String
13333:  Function aUpperCase( const S : String) : String
13334:  Function aLowerCase( const S : string) : String
13335:  Function aCompareText( const S1, S2 : string) : Integer
13336:  Function aSameText( const S1, S2 : string) : Boolean
13337:  Function aInt2Str( Value : Int64) : String
13338:  Function aStr2Int( const Value : String) : Int64
13339:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13340:  Function aGetFileExt( const FileName : String) : String
13341:  Function aGetFilePath( const FileName : String) : String
13342:  Function aGetFileName( const FileName : String) : String
13343:  Function aChangeExt( const FileName, Extension : String) : String
13344:  Function aAdjustLineBreaks( const S : string) : string
13345:  Function aGetWindowStr( WinHandle : HWND) : String
13346:  Function aDiskSpace( Drive : String) : TdriveSize
13347:  Function aFileExists( FileName : String) : Boolean
13348:  Function aFileSize( FileName : String) : Int64
13349:  Function aDirectoryExists( const Name : string) : Boolean
13350:  Function aSysErrorMessage( ErrorCode : Integer) : string
13351:  Function aShortPathName( const LongName : string) : string
13352:  Function aGetWindowVer : TWinVerRec
13353:  procedure InitDriveSpacePtr;
13354: end;
13355:
13356: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13357: begin
13358:  aZero','LongInt'( 0);
13359:  'makeappDEF','LongInt'( - 1);
13360:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13361:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13362:  'CS_KEYCVTWINDOW','LongInt'( 4);
13363:  'CS_DBLCLKS','LongInt'( 8);
```

```
13364:  'CS_OWNDC','LongWord')( $20);
13365:  'CS_CLASSDC','LongWord')( $40);
13366:  'CS_PARENTDC','LongWord')( $80);
13367:  'CS_NOKEYCVT','LongWord')( $100);
13368:  'CS_NOCLOSE','LongWord')( $200);
13369:  'CS_SAVEBITS','LongWord')( $800);
13370:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13371:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13372:  'CS_GLOBALCLASS','LongWord')( $4000);
13373:  'CS_IME','LongWord')( $10000);
13374:  'CS_DROPSHADOW','LongWord')( $20000);
13375:   //PPanelFunc', '^TPanelFunc // will not work
13376:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13377:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13378:   TFontLooks', 'set of TFontLook
13379:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13380:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13381:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13382:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13383:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13384:  Procedure RunMsgLoop( Show : Boolean)
13385:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13386:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13387:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13388:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13389:  Function MakePanel(Left,Top,Width,Height,
       hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13390:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13391:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13392:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13393: end;
13394:
13395: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13396: begin
13397:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13398:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13399:   TScreenSaverOptions', 'set of TScreenSaverOption
13400:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13401:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13402:   SIRegister_TScreenSaver(CL);
13403:  //Procedure Register
13404:  Procedure SetScreenSaverPassword
13405: end;
13406:
13407: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13408: begin
13409:   FindClass('TOBJECT'),'TXCollection
13410:   SIRegister_EFilerException(CL);
13411:   SIRegister_TXCollectionItem(CL);
13412:   //TXCollectionItemClass', 'class of TXCollectionItem
13413:   SIRegister_TXCollection(CL);
13414:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13415:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13416:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13417:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13418:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13419:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13420: end;
13421:
13422: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13423: begin
13424:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13425:  Procedure xglMapTexCoordToNull
13426:  Procedure xglMapTexCoordToMain
13427:  Procedure xglMapTexCoordToSecond
13428:  Procedure xglMapTexCoordToDual
13429:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13430:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13431:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13432:  Procedure xglBeginUpdate
13433:  Procedure xglEndUpdate
13434:  Procedure xglPushState
13435:  Procedure xglPopState
13436:  Procedure xglForbidSecondTextureUnit
13437:  Procedure xglAllowSecondTextureUnit
13438:  Function xglGetBitWiseMapping : Cardinal
13439: end;
13440:
13441: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13442: begin
13443:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13444:   TBaseListOptions', 'set of TBaseListOption
13445:   SIRegister_TBaseList(CL);
13446:   SIRegister_TBaseVectorList(CL);
13447:   SIRegister_TAffineVectorList(CL);
13448:   SIRegister_TVectorList(CL);
13449:   SIRegister_TTexPointList(CL);
```

```
13450:   SIRegister_TXIntegerList(CL);
13451:   //PSingleArrayList', '^TSingleArrayList // will not work
13452:   SIRegister_TSingleList(CL);
13453:   SIRegister_TByteList(CL);
13454:   SIRegister_TQuaternionList(CL);
13455:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13456:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13457:  Procedure FastQuickSortLists(startIndex,
       endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13458: end;
13459:
13460: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13461: begin
13462:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13463:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13464:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
       indices:TIntegerList;list:TAffineVectorList);
13465:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
       indices:TIntegerList;list:TAffineVectorList);
13466:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
       normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13467:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13468:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13469:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13470:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13471:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13472:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13473:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13474:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13475:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13476:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
       edgesTriangles : TIntegerList) : TIntegerList
13477:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13478:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
13479:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13480:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
       TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13481: end;
13482:
13483: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13484: begin
13485:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13486:  Procedure FreeMemAndNil( var P : TObject)
13487:  Function PCharOrNil( const S : string) : PChar
13488:   SIRegister_TJclReferenceMemoryStream(CL);
13489:   FindClass('TOBJECT'),'EJclVMTError
13490:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13491:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13492:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13493:   PDynamicIndexList', '^TDynamicIndexList // will not work
13494:   PDynamicAddressList', '^TDynamicAddressList // will not work
13495:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13496:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13497:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13498:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13499:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13500:  Function GetInitTable( AClass : TClass) : PTypeInfo
13501:   PFieldEntry', '^TFieldEntry // will not work}
13502:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13503:  Function JIsClass( Address : Pointer) : Boolean
13504:  Function JIsObject( Address : Pointer) : Boolean
13505:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13506:   TDigitCount', 'Integer
13507:   SIRegister_TJclNumericFormat(CL);
13508:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13509:   TTextHandler', 'Procedure ( const Text : string)
13510: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13511:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13512:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13513:  Function ReadKey : Char   //to and from the DOS console !
13514:   TModuleHandle', 'HINST
13515:   //TModuleHandle', 'Pointer
13516:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13517:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13518:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13519:  Procedure UnloadModule( var Module : TModuleHandle)
13520:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13521:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13522:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13523:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13524:   FindClass('TOBJECT'),'EJclConversionError
13525:  Function JStrToBoolean( const S : string) : Boolean
13526:  Function JBooleanToStr( B : Boolean) : string
13527:  Function JIntToBool( I : Integer) : Boolean
13528:  Function JBoolToInt( B : Boolean) : Integer
13529:  'ListSeparator','String ';
13530:  'ListSeparator1','String ':
```

```
13531:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13532:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13533:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13534:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13535:  Function ListItemCount( const List, Separator : string) : Integer
13536:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13537:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13538:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13539:  Function SystemTObjectInstance : LongWord
13540:  Function IsCompiledWithPackages : Boolean
13541:  Function JJclGUIDToString( const GUID : TGUID) : string
13542:  Function JJclStringToGUID( const S : string) : TGUID
13543:   SIRegister_TJclIntfCriticalSection(CL);
13544:   SIRegister_TJclSimpleLog(CL);
13545:  Procedure InitSimpleLog( const ALogFileName : string)
13546: end;
13547:
13548: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13549: begin
13550:   FindClass('TOBJECT'),'EJclBorRADException
13551:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13552:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13553:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13554:   TJclBorRADToolPath', 'string
13555:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13556:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13557:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13558:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13559:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13560:  BorRADToolRepositoryFormsPage','String 'Forms
13561:  BorRADToolRepositoryProjectsPage','String 'Projects
13562:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13563:  BorRADToolRepositoryObjectType','String 'Type
13564:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13565:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13566:  BorRADToolRepositoryObjectName','String 'Name
13567:  BorRADToolRepositoryObjectPage','String 'Page
13568:  BorRADToolRepositoryObjectIcon','String 'Icon
13569:  BorRADToolRepositoryObjectDescr','String 'Description
13570:  BorRADToolRepositoryObjectAuthor','String 'Author
13571:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13572:  BorRADToolRepositoryObjectDesigner','String 'Designer
13573:  BorRADToolRepositoryDesignerDfm','String 'dfm
13574:  BorRADToolRepositoryDesignerXfm','String 'xfm
13575:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13576:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13577:  SourceExtensionDelphiPackage','String '.dpk
13578:  SourceExtensionBCBPackage','String '.bpk
13579:  SourceExtensionDelphiProject','String '.dpr
13580:  SourceExtensionBCBProject','String '.bpr
13581:  SourceExtensionBDSProject','String '.bdsproj
13582:  SourceExtensionDProject','String '.dproj
13583:  BinaryExtensionPackage','String '.bpl
13584:  BinaryExtensionLibrary','String '.dll
13585:  BinaryExtensionExecutable','String '.exe
13586:  CompilerExtensionDCP','String '.dcp
13587:  CompilerExtensionBPI','String '.bpi
13588:  CompilerExtensionLIB','String '.lib
13589:  CompilerExtensionTDS','String '.tds
13590:  CompilerExtensionMAP','String '.map
13591:  CompilerExtensionDRC','String '.drc
13592:  CompilerExtensionDEF','String '.def
13593:  SourceExtensionCPP','String '.cpp
13594:  SourceExtensionH','String '.h
13595:  SourceExtensionPAS','String '.pas
13596:  SourceExtensionDFM','String '.dfm
13597:  SourceExtensionXFM','String '.xfm
13598:  SourceDescriptionPAS','String 'Pascal source file
13599:  SourceDescriptionCPP','String 'C++ source file
13600:  DesignerVCL','String 'VCL
13601:  DesignerCLX','String 'CLX
13602:  ProjectTypePackage','String 'package
13603:  ProjectTypeLibrary','String 'library
13604:  ProjectTypeProgram','String 'program
13605:  Personality32Bit','String '32 bit
13606:  Personality64Bit','String '64 bit
13607:  PersonalityDelphi','String 'Delphi
13608:  PersonalityDelphiDotNet','String 'Delphi.net
13609:  PersonalityBCB','String 'C++Builder
13610:  PersonalityCSB','String 'C#Builder
13611:  PersonalityVB','String 'Visual Basic
13612:  PersonalityDesign','String 'Design
13613:  PersonalityUnknown','String 'Unknown personality
13614:  PersonalityBDS','String 'Borland Developer Studio
13615:  DOFDirectoriesSection','String 'Directories
13616:  DOFUnitOutputDirKey','String 'UnitOutputDir
13617:  DOFSearchPathName','String 'SearchPath
13618:  DOFConditionals','String 'Conditionals
13619:  DOFLinkerSection','String 'Linker
```

```
13620:  DOFPackagesKey','String 'Packages
13621:  DOFCompilerSection','String 'Compiler
13622:  DOFPackageNoLinkKey','String 'PackageNoLink
13623:  DOFAdditionalSection','String 'Additional
13624:  DOFOptionsKey','String 'Options
13625:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13626:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13627:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13628:   TJclBorPersonalities', 'set of TJclBorPersonality
13629:   TJclBorDesigner', '( bdVCL, bdCLX )
13630:   TJclBorDesigners', 'set of TJClBorDesigner
13631:   TJclBorPlatform', '( bp32bit, bp64bit )
13632:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13633:   SIRegister_TJclBorRADToolInstallationObject(CL);
13634:   SIRegister_TJclBorlandOpenHelp(CL);
13635:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13636:   TJclHelp2Objects', 'set of TJclHelp2Object
13637:   SIRegister_TJclHelp2Manager(CL);
13638:   SIRegister_TJclBorRADToolIdeTool(CL);
13639:   SIRegister_TJclBorRADToolIdePackages(CL);
13640:   SIRegister_IJclCommandLineTool(CL);
13641:   FindClass('TOBJECT'),'EJclCommandLineToolError
13642:   SIRegister_TJclCommandLineTool(CL);
13643:   SIRegister_TJclBorlandCommandLineTool(CL);
13644:   SIRegister_TJclBCC32(CL);
13645:   SIRegister_TJclDCC32(CL);
13646:   TJclDCC', 'TJclDCC32
13647:   SIRegister_TJclBpr2Mak(CL);
13648:   SIRegister_TJclBorlandMake(CL);
13649:   SIRegister_TJclBorRADToolPalette(CL);
13650:   SIRegister_TJclBorRADToolRepository(CL);
13651:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13652:   TCommandLineTools', 'set of TCommandLineTool
13653:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13654:   SIRegister_TJclBorRADToolInstallation(CL);
13655:   SIRegister_TJclBCBInstallation(CL);
13656:   SIRegister_TJclDelphiInstallation(CL);
13657:   SIRegister_TJclDCCIL(CL);
13658:   SIRegister_TJclBDSInstallation(CL);
13659:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13660:   SIRegister_TJclBorRADToolInstallations(CL);
13661:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13662:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13663:  Function IsDelphiPackage( const FileName : string) : Boolean
13664:  Function IsDelphiProject( const FileName : string) : Boolean
13665:  Function IsBCBPackage( const FileName : string) : Boolean
13666:  Function IsBCBProject( const FileName : string) : Boolean
13667:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13668:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13669:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13670:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
13671:  function SamePath(const Path1, Path2: string): Boolean;
13672: end;
13673:
13674: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13675: begin
13676:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13677:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13678:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13679:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13680:  'LPathSeparator','String '/
13681:  'LDirDelimiter','String '/
13682:  'LDirSeparator','String ':
13683:  'JXPathDevicePrefix','String '\\.\
13684:  'JXPathSeparator','String '\
13685:  'JXDirDelimiter','String '\
13686:  'JXDirSeparator','String ';
13687:  'JXPathUncPrefix','String '\\
13688:  'faNormalFile','LongWord')( $00000080);
13689:  //'faUnixSpecific',' faSymLink);
13690:   JXTCompactPath', '( cpCenter, cpEnd )
13691:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13692:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13693:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13694:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13695:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13696:
13697:  Function jxPathAddSeparator( const Path : string) : string
13698:  Function jxPathAddExtension( const Path, Extension : string) : string
13699:  Function jxPathAppend( const Path, Append : string) : string
13700:  Function jxPathBuildRoot( const Drive : Byte) : string
13701:  Function jxPathCanonicalize( const Path : string) : string
13702:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13703:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13704:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13705:  Function jxPathExtractFileDirFixed( const S : string) : string
13706:  Function jxPathExtractFileNameNoExt( const Path : string) : string
```

```
13707:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13708:  Function jxPathGetDepth( const Path : string) : Integer
13709:  Function jxPathGetLongName( const Path : string) : string
13710:  Function jxPathGetShortName( const Path : string) : string
13711:  Function jxPathGetLongName( const Path : string) : string
13712:  Function jxPathGetShortName( const Path : string) : string
13713:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13714:  Function jxPathGetTempPath : string
13715:  Function jxPathIsAbsolute( const Path : string) : Boolean
13716:  Function jxPathIsChild( const Path, Base : string) : Boolean
13717:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13718:  Function jxPathIsUNC( const Path : string) : Boolean
13719:  Function jxPathRemoveSeparator( const Path : string) : string
13720:  Function jxPathRemoveExtension( const Path : string) : string
13721:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13722:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13723:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13724:   JxTFileListOptions', 'set of TFileListOption
13725:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13726:   TFileHandler', 'Procedure ( const FileName : string)
13727:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13728:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13729:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
      AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
      FileMatchFunc:TFileMatchFunc):Bool;
13730:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13731:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13732:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13733:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13734:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
      RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13735:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
      IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13736:  Procedure jxCreateEmptyFile( const FileName : string)
13737:  Function jxCloseVolume( var Volume : THandle) : Boolean
13738:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13739:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13740:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13741:  Function jxDelTree( const Path : string) : Boolean
13742:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13743:  Function jxDiskInDrive( Drive : Char) : Boolean
13744:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13745:  Function jxFileCreateTemp( var Prefix : string) : THandle
13746:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13747:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13748:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13749:  Function jxFileExists( const FileName : string) : Boolean
13750:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13751:  Function jxFileRestore( const FileName : string) : Boolean
13752:  Function jxGetBackupFileName( const FileName : string) : string
13753:  Function jxIsBackupFileName( const FileName : string) : Boolean
13754:  Function jxFileGetDisplayName( const FileName : string) : string
13755:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13756:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13757:  Function jxFileGetSize( const FileName : string) : Int64
13758:  Function jxFileGetTempName( const Prefix : string) : string
13759:  Function jxFileGetTypeName( const FileName : string) : string
13760:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13761:  Function jxForceDirectories( Name : string) : Boolean
13762:  Function jxGetDirectorySize( const Path : string) : Int64
13763:  Function jxGetDriveTypeStr( const Drive : Char) : string
13764:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13765:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13766:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13767:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13768:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13769:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
      ResolveSymLinks:Boolean):Integer
13770:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13771:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13772:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13773:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13774:  Function jxGetFileCreation( const FName : string) : TFileTime;
13775:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13776:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13777:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13778:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13779:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13780:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13781:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13782:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13783:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13784:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13785:  Function jxGetModulePath( const Module : HMODULE) : string
13786:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13787:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13788:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13789:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13790:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
```

```
13791:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13792:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13793:  Function jxOpenVolume( const Drive : Char) : THandle
13794:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13795:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13796:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13797:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13798:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13799:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13800:  Procedure jxShredFile( const FileName : string; Times : Integer)
13801:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13802:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13803:  Function jxSymbolicLinkTarget( const Name : string) : string
13804:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13805:   SIRegister_TJclCustomFileAttrMask(CL);
13806:   SIRegister_TJclFileAttributeMask(CL);
13807:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13808:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13809:   TFileSearchOptions', 'set of TFileSearchOption
13810:   TFileSearchTaskID', 'Integer
13811:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13812:    +'hTaskID; const Aborted : Boolean)
13813:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13814:   SIRegister_IJclFileEnumerator(CL);
13815:   SIRegister_TJclFileEnumerator(CL);
13816:  Function JxFileSearch : IJclFileEnumerator
13817:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13818:   JxTFileFlags', 'set of TFileFlag
13819:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13820:   SIRegister_TJclFileVersionInfo(CL);
13821:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13822:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13823:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13824:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13825:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13826:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13827:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13828:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13829:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13830:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13831:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13832:   SIRegister_TJclTempFileStream(CL);
13833:   FindClass('TOBJECT'),'TJclCustomFileMapping
13834:   SIRegister_TJclFileMappingView(CL);
13835:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13836:   SIRegister_TJclCustomFileMapping(CL);
13837:   SIRegister_TJclFileMapping(CL);
13838:   SIRegister_TJclSwapFileMapping(CL);
13839:   SIRegister_TJclFileMappingStream(CL);
13840:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13841:   //PPCharArray', '^TPCharArray // will not work
13842:   SIRegister_TJclMappedTextReader(CL);
13843:   SIRegister_TJclFileMaskComparator(CL);
13844:   FindClass('TOBJECT'),'EJclPathError
13845:   FindClass('TOBJECT'),'EJclFileUtilsError
13846:   FindClass('TOBJECT'),'EJclTempFileStreamError
13847:   FindClass('TOBJECT'),'EJclTempFileStreamError
13848:   FindClass('TOBJECT'),'EJclFileMappingError
13849:   FindClass('TOBJECT'),'EJclFileMappingViewError
13850:  Function jxPathGetLongName2( const Path : string) : string
13851:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13852:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13853:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13854:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13855:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13856:  Procedure jxPathListAddItems( var List : string; const Items : string)
13857:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13858:  Procedure jxPathListDelItems( var List : string; const Items : string)
13859:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13860:  Function jxPathListItemCount( const List : string) : Integer
13861:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13862:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13863:  Function jxPathListItemIndex( const List, Item : string) : Integer
13864:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13865:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13866:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13867:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string) : Integer
13868: end;
13869:
13870: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13871: begin
13872:  'UTF8FileHeader','String #$ef#$bb#$bf);
13873:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13874:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13875:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
```

```
13876:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13877:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13878:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13879:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13880:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13881:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13882:  Function lFileIsReadable( const AFilename : string) : boolean
13883:  Function lFileIsWritable( const AFilename : string) : boolean
13884:  Function lFileIsText( const AFilename : string) : boolean
13885:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13886:  Function lFileIsExecutable( const AFilename : string) : boolean
13887:  Function lFileIsSymlink( const AFilename : string) : boolean
13888:  Function lFileIsHardLink( const AFilename : string) : boolean
13889:  Function lFileSize( const Filename : string) : int64;
13890:  Function lGetFileDescription( const AFilename : string) : string
13891:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13892:  Function lTryReadAllLinks( const Filename : string) : string
13893:  Function lDirPathExists( const FileName : String) : Boolean
13894:  Function lForceDirectory( DirectoryName : string) : boolean
13895:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13896:  Function lProgramDirectory : string
13897:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13898:  Function lExtractFileNameOnly( const AFilename : string) : string
13899:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13900:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13901:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13902:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13903:  Function lAppendPathDelim( const Path : string) : string
13904:  Function lChompPathDelim( const Path : string) : string
13905:  Function lTrimFilename( const AFilename : string) : string
13906:  Function lCleanAndExpandFilename( const Filename : string) : string
13907:  Function lCleanAndExpandDirectory( const Filename : string) : string
13908:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13909:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13910:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13911:  Function lFileIsInPath( const Filename, Path : string) : boolean
13912:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13913:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13914:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13915:  'AllDirectoryEntriesMask','String '*
13916:  Function lGetAllFilesMask : string
13917:  Function lGetExeExt : string
13918:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13919:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13920:  Function lFindDiskFilename( const Filename : string) : string
13921:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13922:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13923:  Function lGetDarwinSystemFilename( Filename : string) : string
13924:   SIRegister_TFileIterator(CL);
13925:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13926:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13927:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13928:   SIRegister_TFileSearcher(CL);
13929:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13930:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13931:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13932:  // TCopyFileFlags', 'set of TCopyFileFlag
13933:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13934:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13935:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13936:  Function lReadFileToString( const Filename : string) : string
13937:  Function lGetTempFilename( const Directory, Prefix : string) : string
13938:  {Function NeedRTLAnsi : boolean
13939:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13940:  Function UTF8ToSys( const s : string) : string
13941:  Function SysToUTF8( const s : string) : string
13942:  Function ConsoleToUTF8( const s : string) : string
13943:  Function UTF8ToConsole( const s : string) : string}
13944:  Function FileExistsUTF8( const Filename : string) : boolean
13945:  Function FileAgeUTF8( const FileName : string) : Longint
13946:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13947:  Function ExpandFileNameUTF8( const FileName : string) : string
13948:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13949:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13950:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13951:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13952:  Procedure FindCloseUTF8( var F : TSearchrec)
13953:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13954:  Function FileGetAttrUTF8( const FileName : String) : Longint
13955:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13956:  Function DeleteFileUTF8( const FileName : String) : Boolean
13957:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13958:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13959:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13960:  Function GetCurrentDirUTF8 : String
13961:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
```

```
13962:  Function CreateDirUTF8( const NewDir : String) : Boolean
13963:  Function RemoveDirUTF8( const Dir : String) : Boolean
13964:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13965:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13966:  Function FileCreateUTF8( const FileName : string) : THandle;
13967:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13968:  Function ParamStrUTF8( Param : Integer) : string
13969:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13970:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13971:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13972:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13973:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13974: end;
13975:
13976: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13977: begin
13978:   //VK_F23 = 134;
13979:   //{$EXTERNALSYM VK_F24}
13980:   //VK_F24 = 135;
13981:  TVirtualKeyCode', 'Integer
13982:  'VK_MOUSEWHEELUP','integer'(134);
13983:  'VK_MOUSEWHEELDOWN','integer'(135);
13984:  Function glIsKeyDown( c : Char) : Boolean;
13985:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13986:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13987:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13988:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13989:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13990:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13991: end;
13992:
13993: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13994: begin
13995:   TGLPoint', 'TPoint
13996:   //PGLPoint', '^TGLPoint // will not work
13997:   TGLRect', 'TRect
13998:   //PGLRect', '^TGLRect // will not work
13999:   TDelphiColor', 'TColor
14000:   TGLPicture', 'TPicture
14001:   TGLGraphic', 'TGraphic
14002:   TGLBitmap', 'TBitmap
14003:   //TGraphicClass', 'class of TGraphic
14004:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14005:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14006:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14007:    +'Button; Shift : TShiftState; X, Y : Integer)
14008:   TGLMouseMoveEvent', 'TMouseMoveEvent
14009:   TGLKeyEvent', 'TKeyEvent
14010:   TGLKeyPressEvent', 'TKeyPressEvent
14011:   EGLOSError', 'EWin32Error
14012:   EGLOSError', 'EWin32Error
14013:   EGLOSError', 'EOSError
14014:  'glsAllFilter','string'All // sAllFilter
14015:  Function GLPoint( const x, y : Integer) : TGLPoint
14016:  Function GLRGB( const r, g, b : Byte) : TColor
14017:  Function GLColorToRGB( color : TColor) : TColor
14018:  Function GLGetRValue( rgb : DWORD) : Byte
14019:  Function GLGetGValue( rgb : DWORD) : Byte
14020:  Function GLGetBValue( rgb : DWORD) : Byte
14021:  Procedure GLInitWinColors
14022:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
14023:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
14024:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14025:  Procedure GLInformationDlg( const msg : String)
14026:  Function GLQuestionDlg( const msg : String) : Boolean
14027:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14028:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14029:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14030:  Function GLApplicationTerminated : Boolean
14031:  Procedure GLRaiseLastOSError
14032:  Procedure GLFreeAndNil( var anObject: TObject)
14033:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14034:  Function GLGetCurrentColorDepth : Integer
14035:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14036:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14037:  Procedure GLSleep( length : Cardinal)
14038:  Procedure GLQueryPerformanceCounter( var val : Int64)
14039:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14040:  Function GLStartPrecisionTimer : Int64
14041:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14042:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14043:  Function GLRDTSC : Int64
14044:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14045:  Function GLOKMessageBox( const Text, Caption : string) : Integer
14046:  Procedure GLShowHTMLUrl( Url : String)
14047:  Procedure GLShowCursor( AShow : boolean)
14048:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14049:  Procedure GLGetCursorPos( var point : TGLPoint)
14050:  Function GLGetScreenWidth : integer
```

```
14051:  Function GLGetScreenHeight : integer
14052:  Function GLGetTickCount : int64
14053:  function RemoveSpaces(const str : String) : String;
14054:   TNormalMapSpace','( nmsObject, nmsTangent )
14055:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14056:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14057:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14058:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
14059:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14060: end;
14061:
14062: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14063: begin
14064:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14065:  // PGLStarRecord', '^TGLStarRecord // will not work
14066:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14067:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14068:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14069: end;
14070:
14071:
14072: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14073: begin
14074:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14075:   //PAABB', '^TAABB // will not work
14076:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14077:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14078:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14079:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14080:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14081:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14082:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14083:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14084:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14085:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14086:  Function BBMinX( const c : THmgBoundingBox) : Single
14087:  Function BBMaxX( const c : THmgBoundingBox) : Single
14088:  Function BBMinY( const c : THmgBoundingBox) : Single
14089:  Function BBMaxY( const c : THmgBoundingBox) : Single
14090:  Function BBMinZ( const c : THmgBoundingBox) : Single
14091:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14092:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14093:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14094:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14095:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14096:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14097:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14098:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14099:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14100:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14101:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14102:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14103:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14104:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14105:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14106:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14107:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14108:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14109:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14110:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14111:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14112:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14113:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14114:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14115:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14116:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14117:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14118:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
14119:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14120:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14121:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14122:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14123:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14124:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14125: end;
14126:
14127: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14128: begin
14129:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14130:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14131:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14132:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14133:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14134:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
```

```
14135:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14136:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14137:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14138:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14139:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14140:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14141:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14142:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14143:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14144:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14145:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14146:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14147:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14148:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14149:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14150:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14151:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14152:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14153:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14154: end;
14155:
14156: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14157: begin
14158:  'EPSILON','Single').setExtended( 1e-40);
14159:  'EPSILON2','Single').setExtended( 1e-30);  }
14160:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14161:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14162:  THmgPlane', 'TVector
14163:   TDoubleHmgPlane', 'THomogeneousDblVector
14164:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14165:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14166:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14167:   TSingleArray', 'array of Single
14168:   TTransformations','array [0..15] of Single)
14169:   TPackedRotationMatrix','array [0..2] of Smallint)
14170:   TVertex', 'TAffineVector
14171:   //TVectorGL', 'THomogeneousFltVector
14172:   //TMatrixGL', 'THomogeneousFltMatrix
14173:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14174:  Function glTexPointMake( const s, t : Single) : TTexPoint
14175:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14176:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14177:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14178:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14179:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14180:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14181:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14182:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14183:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14184:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14185:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14186:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14187:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14188:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14189:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14190:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14191:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14192:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14193:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14194:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14195:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14196:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14197:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14198:  Procedure glRstVector( var v : TAffineVector);
14199:  Procedure glRstVector1( var v : TVectorGL);
14200:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14201:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14202:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14203:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14204:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14205:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14206:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14207:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14208:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14209:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14210:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14211:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14212:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Int;dest:PTexPointArray);
14213:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14214:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
      PAffineVectorArray);
14215:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14216:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14217:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14218:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14219:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14220:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
```

```
14221:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14222:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14223:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14224:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14225:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14226:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14227:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14228:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14229:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14230:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14231:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14232:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14233:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14234:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14235:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14236:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14237:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14238:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14239:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14240:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14241:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14242:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14243:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14244:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14245:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14246:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14247:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14248:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14249:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14250:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14251:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14252:  Function glLerp( const start, stop, t : Single) : Single
14253:  Function glAngleLerp( start, stop, t : Single) : Single
14254:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14255:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14256:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14257:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14258:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14259:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14260:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14261:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14262: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14263: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
     PAffineVectorArray);
14264:  Function glVectorLength( const x, y : Single) : Single;
14265:  Function glVectorLength1( const x, y, z : Single) : Single;
14266:  Function glVectorLength2( const v : TAffineVector) : Single;
14267:  Function glVectorLength3( const v : TVectorGL) : Single;
14268:  Function glVectorLength4( const v : array of Single) : Single;
14269:  Function glVectorNorm( const x, y : Single) : Single;
14270:  Function glVectorNorm1( const v : TAffineVector) : Single;
14271:  Function glVectorNorm2( const v : TVectorGL) : Single;
14272:  Function glVectorNorm3( var V : array of Single) : Single;
14273:  Procedure glNormalizeVector( var v : TAffineVector);
14274:  Procedure glNormalizeVector1( var v : TVectorGL);
14275:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14276:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14277: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14278:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14279:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14280:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14281:  Procedure glNegateVector( var V : TAffineVector);
14282:  Procedure glNegateVector2( var V : TVectorGL);
14283:  Procedure glNegateVector3( var V : array of Single);
14284:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14285:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14286:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14287:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14288:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14289:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14290:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14291:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14292:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14293:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14294:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14295:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14296:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14297:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14298:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14299:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14300:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14301:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14302:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14303:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14304:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14305:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14306:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14307:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14308:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
```

```
14309:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14310:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14311:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14312:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14313:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14314:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14315:  Procedure glAbsVector( var v : TVectorGL);
14316:  Procedure glAbsVector1( var v : TAffineVector);
14317:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14318:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14319:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14320:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14321:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14322:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14323:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14324:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14325:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14326:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14327:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14328:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14329:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14330:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14331:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14332:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14333:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14334:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14335:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14336:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14337:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14338:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14339:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14340:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14341:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14342:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14343:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14344:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14345:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14346:  Procedure glAdjointMatrix( var M : TMatrixGL);
14347:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14348:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14349:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14350:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14351:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14352:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14353:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14354:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14355:  Procedure glInvertMatrix( var M : TMatrixGL);
14356:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14357:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14358:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14359:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14360:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14361:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14362:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14363:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14364:  Procedure glNormalizePlane( var plane : THmgPlane)
14365:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14366:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14367:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14368:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14369:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14370:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14371:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14372:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14373:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14374:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14375:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14376:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14377:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14378:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14379:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14380:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14381:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14382:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14383:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14384:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14385:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14386:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14387:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14388:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14389:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14390:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14391:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14392:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14393:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14394:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14395:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14396:  Function glLnXP1( X : Extended) : Extended
```

```
14397:  Function glLog10( X : Extended) : Extended
14398:  Function glLog2( X : Extended) : Extended;
14399:  Function glLog21( X : Single) : Single;
14400:  Function glLogN( Base, X : Extended) : Extended
14401:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14402:  Function glPower( const Base, Exponent : Single) : Single;
14403:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14404:  Function glDegToRad( const Degrees : Extended) : Extended;
14405:  Function glDegToRad1( const Degrees : Single) : Single;
14406:  Function glRadToDeg( const Radians : Extended) : Extended;
14407:  Function glRadToDeg1( const Radians : Single) : Single;
14408:  Function glNormalizeAngle( angle : Single) : Single
14409:  Function glNormalizeDegAngle( angle : Single) : Single
14410:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14411:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14412:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14413:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14414:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14415:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14416:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14417:  Function glArcCos( const X : Extended) : Extended;
14418:  Function glArcCos1( const x : Single) : Single;
14419:  Function glArcSin( const X : Extended) : Extended;
14420:  Function glArcSin1( const X : Single) : Single;
14421:  Function glArcTan21( const Y, X : Extended) : Extended;
14422:  Function glArcTan21( const Y, X : Single) : Single;
14423:  Function glFastArcTan2( y, x : Single) : Single
14424:  Function glTan( const X : Extended) : Extended;
14425:  Function glTan1( const X : Single) : Single;
14426:  Function glCoTan( const X : Extended) : Extended;
14427:  Function glCoTan1( const X : Single) : Single;
14428:  Function glSinh( const x : Single) : Single;
14429:  Function glSinh1( const x : Double) : Double;
14430:  Function glCosh( const x : Single) : Single;
14431:  Function glCosh1( const x : Double) : Double;
14432:  Function glRSqrt( v : Single) : Single
14433:  Function glRLength( x, y : Single) : Single
14434:  Function glISqrt( i : Integer) : Integer
14435:  Function glILength( x, y : Integer) : Integer;
14436:  Function glILength1( x, y, z : Integer) : Integer;
14437:  Procedure glRegisterBasedExp
14438:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14439:  Function glRoundInt( v : Single) : Single;
14440:  Function glRoundInt1( v : Extended) : Extended;
14441:  Function glTrunc( v : Single) : Integer;
14442:  Function glTrunc64( v : Extended) : Int64;
14443:  Function glInt( v : Single) : Single;
14444:  Function glInt1( v : Extended) : Extended;
14445:  Function glFrac( v : Single) : Single;
14446:  Function glFrac1( v : Extended) : Extended;
14447:  Function glRound( v : Single) : Integer;
14448:  Function glRound64( v : Single) : Int64;
14449:  Function glRound641( v : Extended) : Int64;
14450:  Function glTrunc( X : Extended) : Int64
14451:  Function glRound( X : Extended) : Int64
14452:  Function glFrac( X : Extended) : Extended
14453:  Function glCeil( v : Single) : Integer;
14454:  Function glCeil64( v : Extended) : Int64;
14455:  Function glFloor( v : Single) : Integer;
14456:  Function glFloor64( v : Extended) : Int64;
14457:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14458:  Function glSign( x : Single) : Integer
14459:  Function glIsInRange( const x, a, b : Single) : Boolean;
14460:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14461:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14462:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14463:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14464:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14465:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14466:  Function glMinFloat3( const v1, v2 : Single) : Single;
14467:  Function glMinFloat4( const v : array of Single) : Single;
14468:  Function glMinFloat5( const v1, v2 : Double) : Double;
14469:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14470:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14471:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14472:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14473:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14474:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14475:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14476:  Function glMaxFloat2( const v : array of Single) : Single;
14477:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14478:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14479:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14480:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14481:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14482:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14483:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14484:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14485:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
```

```
14486:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14487:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14488:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14489:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14490:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14491:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14492:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14493:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14494:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14495:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14496:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14497:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14498:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14499:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14500:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14501:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14502:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14503:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14504:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14505:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14506:  Procedure glSortArrayAscending( var a : array of Extended)
14507:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14508:  Function glClampValue1( const aValue, aMin : Single) : Single;
14509:  Function glGeometryOptimizationMode : String
14510:  Procedure glBeginFPUOnlySection
14511:  Procedure glEndFPUOnlySection
14512:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14513:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14514:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14515:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14516:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14517:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14518:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14519:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14520:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14521:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14522:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14523:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14524:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14525:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14526:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14527:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14528:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14529:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14530:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14531:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14532:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14533:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14534:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14535:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14536:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14537:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14538:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14539:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14540:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14541:  'cPI','Single').setExtended( 3.141592654);
14542:  'cPIdiv180','Single').setExtended( 0.017453292);
14543:  'c180divPI','Single').setExtended( 57.29577951);
14544:  'c2PI','Single').setExtended( 6.283185307);
14545:  'cPIdiv2','Single').setExtended( 1.570796326);
14546:  'cPIdiv4','Single').setExtended( 0.785398163);
14547:  'c3PIdiv4','Single').setExtended( 2.35619449);
14548:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14549:  'cInv360','Single').setExtended( 1 / 360);
14550:  'c180','Single').setExtended( 180);
14551:  'c360','Single').setExtended( 360);
14552:  'cOneHalf','Single').setExtended( 0.5);
14553:  'cLn10','Single').setExtended( 2.302585093);
14554:  {'MinSingle','Extended').setExtended( 1.5e-45);
14555:  'MaxSingle','Extended').setExtended( 3.4e+38);
14556:  'MinDouble','Extended').setExtended( 5.0e-324);
14557:  'MaxDouble','Extended').setExtended( 1.7e+308);
14558:  'MinExtended','Extended').setExtended( 3.4e-4932);
14559:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14560:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14561:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14562: end;
14563:
14564: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14565: begin
14566:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14567:   (FindClass('TOBJECT'),'TFaceGroups
14568:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14569:   TMeshAutoCenterings', 'set of TMeshAutoCentering
```

```
14570:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14571:   SIRegister_TBaseMeshObject(CL);
14572:   (FindClass('TOBJECT'),'TSkeletonFrameList
14573:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14574:   SIRegister_TSkeletonFrame(CL);
14575:   SIRegister_TSkeletonFrameList(CL);
14576:   (FindClass('TOBJECT'),'TSkeleton
14577:   (FindClass('TOBJECT'),'TSkeletonBone
14578:   SIRegister_TSkeletonBoneList(CL);
14579:   SIRegister_TSkeletonRootBoneList(CL);
14580:   SIRegister_TSkeletonBone(CL);
14581:   (FindClass('TOBJECT'),'TSkeletonColliderList
14582:   SIRegister_TSkeletonCollider(CL);
14583:   SIRegister_TSkeletonColliderList(CL);
14584:   (FindClass('TOBJECT'),'TGLBaseMesh
14585:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14586:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14587:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14588:    +'QuaternionList; end
14589:   SIRegister_TSkeleton(CL);
14590:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14591:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14592:   SIRegister_TMeshObject(CL);
14593:   SIRegister_TMeshObjectList(CL);
14594:   //TMeshObjectListClass', 'class of TMeshObjectList
14595:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14596:   SIRegister_TMeshMorphTarget(CL);
14597:   SIRegister_TMeshMorphTargetList(CL);
14598:   SIRegister_TMorphableMeshObject(CL);
14599:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14600:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14601:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14602:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14603:   SIRegister_TSkeletonMeshObject(CL);
14604:   SIRegister_TFaceGroup(CL);
14605:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14606:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14607:   SIRegister_TFGVertexIndexList(CL);
14608:   SIRegister_TFGVertexNormalTexIndexList(CL);
14609:   SIRegister_TFGIndexTexCoordList(CL);
14610:   SIRegister_TFaceGroups(CL);
14611:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14612:   SIRegister_TVectorFile(CL);
14613:   //TVectorFileClass', 'class of TVectorFile
14614:   SIRegister_TGLGLSMVectorFile(CL);
14615:   SIRegister_TGLBaseMesh(CL);
14616:   SIRegister_TGLFreeForm(CL);
14617:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14618:   TGLActorOptions', 'set of TGLActorOption
14619:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14620:   (FindClass('TOBJECT'),'TGLActor
14621:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14622:   SIRegister_TActorAnimation(CL);
14623:   TActorAnimationName', 'String
14624:   SIRegister_TActorAnimations(CL);
14625:   SIRegister_TGLBaseAnimationControler(CL);
14626:   SIRegister_TGLAnimationControler(CL);
14627:   TActorFrameInterpolation', '( afpNone, afpLinear )
14628:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14629:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14630:   SIRegister_TGLActor(CL);
14631:   SIRegister_TVectorFileFormat(CL);
14632:   SIRegister_TVectorFileFormatsList(CL);
14633:   (FindClass('TOBJECT'),'EInvalidVectorFile
14634:  Function GetVectorFileFormats : TVectorFileFormatsList
14635:  Function VectorFileFormatsFilter : String
14636:  Function VectorFileFormatsSaveFilter : String
14637:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14638:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14639:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14640: end;
14641:
14642: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14643: begin
14644:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14645:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14646:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14647:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14648:   SIRegister_TOleStream(CL);
14649:   (FindClass('TOBJECT'),'TConnectionPoints
14650:   TConnectionKind', '( ckSingle, ckMulti )
14651:   SIRegister_TConnectionPoint(CL);
14652:   SIRegister_TConnectionPoints(CL);
14653:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14654:   (FindClass('TOBJECT'),'TActiveXControlFactory
14655:   SIRegister_TActiveXControl(CL);
14656:   //TActiveXControlClass', 'class of TActiveXControl
14657:   SIRegister_TActiveXControlFactory(CL);
14658:   SIRegister_TActiveFormControl(CL);
```

```
14659:   SIRegister_TActiveForm(CL);
14660:   //TActiveFormClass', 'class of TActiveForm
14661:   SIRegister_TActiveFormFactory(CL);
14662:   (FindClass('TOBJECT'),'TPropertyPageImpl
14663:   SIRegister_TPropertyPage(CL);
14664:   //TPropertyPageClass', 'class of TPropertyPage
14665:   SIRegister_TPropertyPageImpl(CL);
14666:   SIRegister_TActiveXPropertyPage(CL);
14667:   SIRegister_TActiveXPropertyPageFactory(CL);
14668:   SIRegister_TCustomAdapter(CL);
14669:   SIRegister_TAdapterNotifier(CL);
14670:   SIRegister_IFontAccess(CL);
14671:   SIRegister_TFontAdapter(CL);
14672:   SIRegister_IPictureAccess(CL);
14673:   SIRegister_TPictureAdapter(CL);
14674:   SIRegister_TOleGraphic(CL);
14675:   SIRegister_TStringsAdapter(CL);
14676:   SIRegister_TReflectorWindow(CL);
14677:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14678:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14679:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14680:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14681:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14682:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14683:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14684:  Function ParkingWindow : HWND
14685: end;
14686:
14687: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14688: begin
14689:  // TIp6Bytes = array [0..15] of Byte;
14690: {:binary form of IPv6 adress (for string conversion routines)}
14691:  // TIp6Words = array [0..7] of Word;
14692:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14693:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14694:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14695:  Function synaIsIP( const Value : string) : Boolean');
14696:  Function synaIsIP6( const Value : string) : Boolean');
14697:  Function synaIPToID( Host : string) : Ansistring');
14698:  Function synaStrToIp6( value : string) : TIp6Bytes');
14699:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14700:  Function synaStrToIp( value : string) : integer');
14701:  Function synaIpToStr( value : integer) : string');
14702:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14703:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14704:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14705:  Function xStrToIP( const Value : String) : TIPAdr');
14706:  Function xIPToStr( const Adresse : TIPAdr) : String');
14707:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14708:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14709: end;
14710:
14711: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14712: begin
14713:  AddTypeS('TSpecials', 'set of Char');
14714:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14715:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14716:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14717:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14718:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14719:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14720:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14721:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14722:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14723:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14724:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14725:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14726:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14727:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14728:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14729:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14730:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14731:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14732:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14733:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14734:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14735:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14736:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14737:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14738:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14739:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14740:  Function synCrc32( const Value : AnsiString) : Integer');
14741:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14742:  Function Crc16( const Value : AnsiString) : Word');
14743:  Function synMD5( const Value : AnsiString) : AnsiString');
14744:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14745:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14746:  Function synSHA1( const Value : AnsiString) : AnsiString');
14747:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
```

```
14748:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14749:  Function synMD4( const Value : AnsiString) : AnsiString');
14750: end;
14751:
14752: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14753: begin
14754:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14755:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14756:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14757:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14758:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14759:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14760:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14761:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14762:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14763:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14764:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14765:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14766:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14767:    +', CP864, CP865, CP869, CP1125 )');
14768:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14769:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14770:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
       TransformTable : array of Word) : AnsiString');
14771:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
       TransformTable : array of Word; Translit : Boolean) : AnsiString');
14772:  Function GetCurCP : TMimeChar');
14773:  Function GetCurOEMCP : TMimeChar');
14774:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14775:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14776:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14777:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14778:  Function GetBOM( Value : TMimeChar) : AnsiString');
14779:  Function StringToWide( const Value : AnsiString) : WideString');
14780:  Function WideToString( const Value : WideString) : AnsiString');
14781: end;
14782:
14783: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14784: begin
14785:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14786:  Procedure WakeOnLan( MAC, IP : string)');
14787:  Function GetDNS : string');
14788:  Function GetIEProxy( protocol : string) : TProxySetting');
14789:  Function GetLocalIPs : string');
14790: end;
14791:
14792:
14793: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14794: begin
14795:  AddConstantN('synCR','Char #$0d);
14796:  Const('synLF','Char #$0a);
14797:  Const('cSerialChunk','LongInt'( 8192);
14798:  Const('LockfileDirectory','String '/var/lock');
14799:  Const('PortIsClosed','LongInt'( - 1);
14800:  Const('ErrAlreadyOwned','LongInt'( 9991);
14801:  Const('ErrAlreadyInUse','LongInt'( 9992);
14802:  Const('ErrWrongParameter','LongInt'( 9993);
14803:  Const('ErrPortNotOpen','LongInt'( 9994);
14804:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14805:  Const('ErrMaxBuffer','LongInt'( 9996);
14806:  Const('ErrTimeout','LongInt'( 9997);
14807:  Const('ErrNotRead','LongInt'( 9998);
14808:  Const('ErrFrame','LongInt'( 9999);
14809:  Const('ErrOverrun','LongInt'( 10000);
14810:  Const('ErrRxOver','LongInt'( 10001);
14811:  Const('ErrRxParity','LongInt'( 10002);
14812:  Const('ErrTxFull','LongInt'( 10003);
14813:  Const('dcb_Binary','LongWord')( $00000001);
14814:  Const('dcb_ParityCheck','LongWord')( $00000002);
14815:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14816:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14817:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14818:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14819:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14820:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14821:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14822:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14823:  Const('dcb_OutX','LongWord')( $00000100);
14824:  Const('dcb_InX','LongWord')( $00000200);
14825:  Const('dcb_ErrorChar','LongWord')( $00000400);
14826:  Const('dcb_NullStrip','LongWord')( $00000800);
14827:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14828:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14829:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14830:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14831:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14832:  Const('dcb_AbortOnError','LongWord')( $00004000);
14833:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14834:  Const('synSB1','LongInt'( 0);
```

```
14835:  Const('SB1andHalf','LongInt'( 1);
14836:  Const('synSB2','LongInt'( 2);
14837:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14838:  Const('CS7fix','LongWord')( $0000020);
14839:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14840:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14841:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14842:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14843:   //AddTypeS('PDCB', '^TDCB // will not work');
14844:  //Const('MaxRates','LongInt'( 18);
14845:  //Const('MaxRates','LongInt'( 30);
14846:  //Const('MaxRates','LongInt'( 19);
14847:  Const('O_SYNC','LongWord')( $0080);
14848:  Const('synOK','LongInt'( 0);
14849:  Const('synErr','LongInt'( integer ( - 1 ));
14850:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14851:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14852:   SIRegister_ESynaSerError(CL);
14853:   SIRegister_TBlockSerial(CL);
14854:  Function GetSerialPortNames : string');
14855: end;
14856:
14857: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14858: begin
14859:  Const('DLLIconvName','String 'libiconv.so');
14860:  Const('DLLIconvName','String 'iconv.dll');
14861:   AddTypeS('size_t', 'Cardinal');
14862:   AddTypeS('iconv_t', 'Integer');
14863:   //AddTypeS('iconv_t', 'Pointer');
14864:   AddTypeS('argptr', 'iconv_t');
14865:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14866:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14867:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14868:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14869:  Function SynaIconvClose( var cd : iconv_t) : integer');
14870:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14871:  Function IsIconvloaded : Boolean');
14872:  Function InitIconvInterface : Boolean');
14873:  Function DestroyIconvInterface : Boolean');
14874:  Const('ICONV_TRIVIALP','LongInt'( 0);
14875:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14876:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14877:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14878:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14879: end;
14880:
14881: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14882: begin
14883:  Const 'ICMP_ECHO','LongInt'( 8);
14884:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14885:  Const('ICMP_UNREACH','LongInt'( 3);
14886:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14887:  Const('ICMP6_ECHO','LongInt'( 128);
14888:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14889:  Const('ICMP6_UNREACH','LongInt'( 1);
14890:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14891:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14892:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14893:   SIRegister_TPINGSend(CL);
14894:  Function PingHost( const Host : string) : Integer');
14895:  Function TraceRouteHost( const Host : string) : string');
14896: end;
14897:
14898: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14899: begin
14900:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14901:  Const('synASN1_INT','LongWord')( $02);
14902:  Const('synASN1_OCTSTR','LongWord')( $04);
14903:  Const('synASN1_NULL','LongWord')( $05);
14904:  Const('synASN1_OBJID','LongWord')( $06);
14905:  Const('synASN1_ENUM','LongWord')( $0a);
14906:  Const('synASN1_SEQ','LongWord')( $30);
14907:  Const('synASN1_SETOF','LongWord')( $31);
14908:  Const('synASN1_IPADDR','LongWord')( $40);
14909:  Const('synASN1_COUNTER','LongWord')( $41);
14910:  Const('synASN1_GAUGE','LongWord')( $42);
14911:  Const('synASN1_TIMETICKS','LongWord')( $43);
14912:  Const('synASN1_OPAQUE','LongWord')( $44);
14913:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14914:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14915:  Function synASNEncLen( Len : Integer) : AnsiString');
14916:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14917:  Function synASNEncInt( Value : Integer) : AnsiString');
14918:  Function synASNEncUInt( Value : Integer) : AnsiString');
14919:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14920:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14921:  Function synMibToId( Mib : String) : AnsiString');
14922:  Function synIdToMib( const Id : AnsiString) : String');
```

```
14923:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14924:  Function ASNdump( const Value : AnsiString) : AnsiString');
14925:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14926:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14927: end;
14928:
14929: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14930: begin
14931:  Const('cLDAPProtocol','String '389');
14932:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14933:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14934:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14935:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14936:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14937:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14938:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14939:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14940:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14941:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14942:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14943:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14944:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14945:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14946:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14947:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14948:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14949:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14950:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14951:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14952:   SIRegister_TLDAPAttribute(CL);
14953:   SIRegister_TLDAPAttributeList(CL);
14954:   SIRegister_TLDAPResult(CL);
14955:   SIRegister_TLDAPResultList(CL);
14956:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14957:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14958:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14959:   SIRegister_TLDAPSend(CL);
14960:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14961: end;
14962:
14963:
14964: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14965: begin
14966:  Const('cSysLogProtocol','String '514');
14967:  Const('FCL_Kernel','LongInt'( 0);
14968:  Const('FCL_UserLevel','LongInt'( 1);
14969:  Const('FCL_MailSystem','LongInt'( 2);
14970:  Const('FCL_System','LongInt'( 3);
14971:  Const('FCL_Security','LongInt'( 4);
14972:  Const('FCL_Syslogd','LongInt'( 5);
14973:  Const('FCL_Printer','LongInt'( 6);
14974:  Const('FCL_News','LongInt'( 7);
14975:  Const('FCL_UUCP','LongInt'( 8);
14976:  Const('FCL_Clock','LongInt'( 9);
14977:  Const('FCL_Authorization','LongInt'( 10);
14978:  Const('FCL_FTP','LongInt'( 11);
14979:  Const('FCL_NTP','LongInt'( 12);
14980:  Const('FCL_LogAudit','LongInt'( 13);
14981:  Const('FCL_LogAlert','LongInt'( 14);
14982:  Const('FCL_Time','LongInt'( 15);
14983:  Const('FCL_Local0','LongInt'( 16);
14984:  Const('FCL_Local1','LongInt'( 17);
14985:  Const('FCL_Local2','LongInt'( 18);
14986:  Const('FCL_Local3','LongInt'( 19);
14987:  Const('FCL_Local4','LongInt'( 20);
14988:  Const('FCL_Local5','LongInt'( 21);
14989:  Const('FCL_Local6','LongInt'( 22);
14990:  Const('FCL_Local7','LongInt'( 23);
14991:   Type(TSyslogSeverity','(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14992:   SIRegister_TSyslogMessage(CL);
14993:   SIRegister_TSyslogSend(CL);
14994:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
      Content:string):Boolean;
14995: end;
14996:
14997:
14998: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14999: begin
15000:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15001:   SIRegister_TMessHeader(CL);
15002:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
15003:   SIRegister_TMimeMess(CL);
15004: end;
15005:
15006: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
15007: begin
15008:   (FindClass('TOBJECT'),'TMimePart');
15009:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15010:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
```

```
15011:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15012:   SIRegister_TMimePart(CL);
15013:  Const('MaxMimeType','LongInt'( 25);
15014:  Function GenerateBoundary : string');
15015: end;
15016:
15017: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15018: begin
15019:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
15020:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
15021:  Function NeedInline( const Value : AnsiString) : boolean');
15022:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15023:  Function InlineCode( const Value : string) : string');
15024:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15025:  Function InlineEmail( const Value : string) : string');
15026: end;
15027:
15028: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15029: begin
15030:  Const('cFtpProtocol','String '21');
15031:  Const('cFtpDataProtocol','String '20');
15032:  Const('FTP_OK','LongInt'( 255);
15033:  Const('FTP_ERR','LongInt'( 254);
15034:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15035:   SIRegister_TFTPListRec(CL);
15036:   SIRegister_TFTPList(CL);
15037:   SIRegister_TFTPSend(CL);
15038:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15039:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15040:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
      ToPort, ToFile, ToUser, ToPass : string) : Boolean');
15041: end;
15042:
15043: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15044: begin
15045:  Const('cHttpProtocol','String '80');
15046:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15047:   SIRegister_THTTPSend(CL);
15048:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
15049:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
15050:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
15051:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
15052:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
      ResultData:TStrings):Bool;
15053: end;
15054:
15055: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15056: begin
15057:  Const('cSmtpProtocol','String '25');
15058:   SIRegister_TSMTPSend(CL);
15059:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
      Passw:string):Bool;
15060:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15061:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
      Username, Password : string):Boolean');
15062: end;
15063:
15064: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15065: begin
15066:  Const('cSnmpProtocol','String '161');
15067:  Const('cSnmpTrapProtocol','String '162');
15068:  Const('SNMP_V1','LongInt'( 0);
15069:  Const('SNMP_V2C','LongInt'( 1);
15070:  Const('SNMP_V3','LongInt'( 3);
15071:  Const('PDUGetRequest','LongWord')( $A0);
15072:  Const('PDUGetNextRequest','LongWord')( $A1);
15073:  Const('PDUGetResponse','LongWord')( $A2);
15074:  Const('PDUSetRequest','LongWord')( $A3);
15075:  Const('PDUTrap','LongWord')( $A4);
15076:  Const('PDUGetBulkRequest','LongWord')( $A5);
15077:  Const('PDUInformRequest','LongWord')( $A6);
15078:  Const('PDUTrapV2','LongWord')( $A7);
15079:  Const('PDUReport','LongWord')( $A8);
15080:  Const('ENoError',LongInt 0;
15081:  Const('ETooBig','LongInt')( 1);
15082:  Const('ENoSuchName','LongInt'( 2);
15083:  Const('EBadValue','LongInt'( 3);
15084:  Const('EReadOnly','LongInt'( 4);
15085:  Const('EGenErr','LongInt'( 5);
15086:  Const('ENoAccess','LongInt'( 6);
15087:  Const('EWrongType','LongInt'( 7);
15088:  Const('EWrongLength','LongInt'( 8);
15089:  Const('EWrongEncoding','LongInt'( 9);
15090:  Const('EWrongValue','LongInt'( 10);
15091:  Const('ENoCreation','LongInt'( 11);
15092:  Const('EInconsistentValue','LongInt'( 12);
15093:  Const('EResourceUnavailable','LongInt'( 13);
15094:  Const('ECommitFailed','LongInt'( 14);
15095:  Const('EUndoFailed','LongInt'( 15);
```

```
15096:  Const('EAuthorizationError','LongInt'( 16);
15097:  Const('ENotWritable','LongInt'( 17);
15098:  Const('EInconsistentName','LongInt'( 18);
15099:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15100:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15101:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15102:   SIRegister_TSNMPMib(CL);
15103:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15104:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15105:   SIRegister_TSNMPRec(CL);
15106:   SIRegister_TSNMPSend(CL);
15107:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15108:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15109:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15110:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15111:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15112:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15113:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : TStringList) : Integer');
15114: end;
15115:
15116: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15117: begin
15118:  Function GetDomainName2: AnsiString');
15119:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15120:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15121:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15122:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15123:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15124: end;
15125:
15126: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15127: begin
15128:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15129:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15130:  Function wwStrToDate( const S : string) : boolean');
15131:  Function wwStrToTime( const S : string) : boolean');
15132:  Function wwStrToDateTime( const S : string) : boolean');
15133:  Function wwStrToTimeVal( const S : string) : TDateTime');
15134:  Function wwStrToDateVal( const S : string) : TDateTime');
15135:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15136:  Function wwStrToInt( const S : string) : boolean');
15137:  Function wwStrToFloat( const S : string) : boolean');
15138:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15139:  Function wwNextDay( Year, Month, Day : Word) : integer');
15140:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15141:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15142:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15143:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15144:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15145:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15146:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15147:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15148:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15149: end;
15150:
15151: unit uPSI_Themes;
15152: Function ThemeServices : TThemeServices');
15153: Function ThemeControl( AControl : TControl) : Boolean');
15154: Function UnthemedDesigner( AControl : TControl) : Boolean');
15155: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15156: begin
15157:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15158: end;
15159: Unit uPSC_menus;
15160:  Function StripHotkey( const Text : string) : string');
15161:  Function GetHotkey( const Text : string) : string');
15162:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15163:  Function IsAltGRPressed : boolean');
15164:
15165: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15166: begin
15167:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15168:   SIRegister_TIdIMAP4Server(CL);
15169: end;
15170:
15171: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15172: begin
15173:  'HASH_SIZE','LongInt'( 256);
15174:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15175:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15176:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15177:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15178:    +' : Integer; Value : Variant; end');
15179:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15180:   SIRegister_TVariantSymbolTable(CL);
15181: end;
15182:
```

```
15183: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15184: begin
15185:   SIRegister_TThreadLocalVariables(CL);
15186:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15187:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15188:  Function ThreadLocals : TThreadLocalVariables');
15189:  Procedure WriteDebug( sz : String)');
15190:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15191:  'UDF_FAILURE','LongInt'( 1);
15192:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15193:  CL.AddTypeS('mTByteArray', 'array of byte;');
15194:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15195:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15196:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15197:  function IsNetworkConnected: Boolean;
15198:  function IsInternetConnected: Boolean;
15199:  function IsCOMConnected: Boolean;
15200:  function IsNetworkOn: Boolean;
15201:  function IsInternetOn: Boolean;
15202:  function IsCOMOn: Boolean;
15203:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15204:  TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15205:  Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT');
15206:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15207:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15208:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15209:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15210:  Function GetMenu( hWnd : HWND) : HMENU');
15211:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15212: end;
15213:
15214: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15215: begin
15216:   SIRegister_IDataBlock(CL);
15217:   SIRegister_ISendDataBlock(CL);
15218:   SIRegister_ITransport(CL);
15219:   SIRegister_TDataBlock(CL);
15220:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15221:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15222:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15223:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15224:   SIRegister_TCustomDataBlockInterpreter(CL);
15225:   SIRegister_TSendDataBlock(CL);
15226:   'CallSig','LongWord')( $D800);
15227:   'ResultSig','LongWord')( $D400);
15228:   'asMask','LongWord')( $00FF);
15229:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15230:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15231:  Procedure CheckSignature( Sig : Integer)');
15232: end;
15233:
15234: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15235: begin
15236:   //CL.AddTypeS('HINTERNET', '___Pointer');
15237:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15238:   CL.AddTypeS('HINTERNET', 'Integer');
15239:   CL.AddTypeS('HINTERNET2', '___Pointer');
15240:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15241:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15242:   CL.AddTypeS('INTERNET_PORT', 'Word');
15243:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15244:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15245:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15246:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15247:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15248:  Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
       lpUrlComponents:TURLComponents):BOOL;
15249:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
       dwUrlLength:DWORD):BOOL;
15250:  Function InternetCloseHandle(hInet : HINTERNET) : BOOL');
15251:  Function
       InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
       lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15252:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15253:  ;dwContext:DWORD):HINTERNET;
15254:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
       lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15255:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
       dwContext:DWORD):BOOL;
15256:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15257:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15258:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15259:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15260:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15261:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15262:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15263:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
       lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
```

```
15264:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
       lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15265:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15266:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15267:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
       dwContext:DWORD):BOOL;
15268:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
       TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15269:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
       BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15270:  Function
       WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15271:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15272:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15273:  Function
       FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15274:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15275:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15276:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15277:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15278:  Function
       FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15279:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15280:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15281:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15282:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15283:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15284:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15285:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15286:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15287:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15288:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15289:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15290:  Function
       GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
       PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15291:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15292:  Function
       GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15293:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
       PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15294:  Function
       HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15295:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
       dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15296:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15297:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15298:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15299:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15300:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15301:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15302:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15303:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
       lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15304:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
       lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15305:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15306:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15307:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15308:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15309: end;
15310:
15311: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15312: begin
15313:   AddTypeS('strCharSet', 'set of char');
15314:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15315:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15316:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15317:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15318:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15319:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15320:  Procedure strStripWhiteSpace( var s : string)');
15321:  Function strRemoveChar( str : string; removeChar : char) : string');
15322:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15323:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15324:  Function wwEqualStr( s1, s2 : string) : boolean');
15325:  Function strCount( s : string; delimeter : char) : integer');
15326:  Function strWhiteSpace : strCharSet');
15327:  Function wwExtractFileNameOnly( const FileName : string) : string');
15328:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15329:  Function strTrailing( s : string; delimeter : char) : string');
15330:  Function strPreceding( s : string; delimeter : char) : string');
15331:  Function wwstrReplace( s, Find, Replace : string) : string');
15332: end;
15333:
15334: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15335: begin
15336:   SIRegister_TRemoteDataModule(CL);
```

```
15337:   SIRegister_TCRemoteDataModule(CL);
15338:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15339:  Procedure UnregisterPooled( const ClassID : string)');
15340:  Procedure EnableSocketTransport( const ClassID : string)');
15341:  Procedure DisableSocketTransport( const ClassID : string)');
15342:  Procedure EnableWebTransport( const ClassID : string)');
15343:  Procedure DisableWebTransport( const ClassID : string)');
15344: end;
15345:
15346: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15347: begin
15348:  Function mxArcCos( x : Real) : Real');
15349:  Function mxArcSin( x : Real) : Real');
15350:  Function Comp2Str( N : Comp) : String');
15351:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15352:  Function Int2Str( N : LongInt) : String');
15353:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15354:  Function LogXY( x, y : Real) : Real');
15355:  Function Pennies2Dollars( C : Comp) : String');
15356:  Function mxPower( X : Integer; Y : Integer) : Real');
15357:  Function Real2Str( N : Real; Width, Places : integer) : String');
15358:  Function mxStr2Comp( MyString : string) : Comp');
15359:  Function mxStr2Pennies( S : String) : Comp');
15360:  Function Str2Real( MyString : string) : Real');
15361:  Function XToTheY( x, y : Real) : Real');
15362: end;
15363:
15364: //************************Cindy Functions!*****************************
15365: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15366: begin
15367:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15368:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15369:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15370:  MessagePlainText','String 'text/plain');
15371:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15372:  MessageAlterText_Html','String 'multipart/alternative');
15373:  MessageHtml_Attach','String 'multipart/mixed');
15374:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15375:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15376:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15377:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15378:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15379:  Function ForceDecodeHeader( aHeader : String) : String');
15380:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15381:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15382:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15383:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15384:  Function Get_MD5( const aFileName : string) : string');
15385:  Function Get_MD5FromString( const aString : string) : string');
15386: end;
15387:
15388: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15389: begin
15390:  Function IsFolder( SRec : TSearchrec) : Boolean');
15391:  Function isFolderReadOnly( Directory : String) : Boolean');
15392:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15393:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15394:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15395:  Function DiskFreeBytes( Drv : Char) : Int64');
15396:  Function DiskBytes( Drv : Char) : Int64');
15397:  Function GetFileBytes( Filename : String) : Int64');
15398:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15399:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15400:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15401:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15402:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15403:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15404:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15405:  SE_TCB_NAME','String 'SeTcbPrivilege');
15406:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15407:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15408:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15409:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15410:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15411:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15412:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15413:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15414:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15415:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15416:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15417:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15418:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15419:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15420:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15421:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15422:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15423:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15424:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15425:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
```

```
15426:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15427: end;
15428:
15429:
15430: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15431: begin
15432:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15433:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15434:  Function ShellGetExtensionName( FileName : String) : String');
15435:  Function ShellGetIconIndex( FileName : String) : Integer');
15436:  Function ShellGetIconHandle( FileName : String) : HIcon');
15437:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15438:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15439:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15440:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15441:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15442:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15443:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15444:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
      WaitCloseCompletion : boolean) : Boolean');
15445:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15446:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15447:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15448:  Function GetModificationDate( Filename : String) : TDateTime');
15449:  Function GetCreationDate( Filename : String) : TDateTime');
15450:  Function GetLastAccessDate( Filename : String) : TDateTime');
15451:  Function FileDelete( Filename : String) : Boolean');
15452:  Function FileIsOpen( Filename : string) : boolean');
15453:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15454:  Function DirectoryDelete( Directory : String) : Boolean');
15455:  Function GetPrinters( PrintersList : TStrings) : Integer');
15456:  Procedure SetDefaultPrinter( PrinterName : String)');
15457:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15458:  Function WinToDosPath( WinPathName : String) : String');
15459:  Function DosToWinPath( DosPathName : String) : String');
15460:  Function cyGetWindowsVersion : TWindowsVersion');
15461:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15462:  Procedure WindowsShutDown( Restart : boolean)');
15463:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
      FileIcon:string;NumIcone:integer);
15464:  Procedure GetWindowsFonts( FontsList : TStrings)');
15465:  Function GetAvailableFilename( DesiredFileName : String): String');
15466: end;
15467:
15468: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15469: begin
15470:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15471:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15472:   Type(TStringRead', '( srFromLeft, srFromRight )');
15473:   Type(TStringReads', 'set of TStringRead');
15474:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15475:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15476:   Type(TWordsOptions', 'set of TWordsOption');
15477:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15478:   Type(TCarTypes', 'set of TCarType');
15479:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15480:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15481:  Function Char_GetType( aChar : Char) : TCarType');
15482:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15483:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15484:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15485:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15486:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15487:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15488:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15489:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15490:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15491:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15492:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15493:  Function String_Quote( Str : String) : String');
15494:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15495:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15496:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15497:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15498:  Function String_ToInt( Str : String) : Integer');
15499:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15500:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15501:  Function String_Reverse( Str : String) : String');
15502:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15503:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
      TCaseSensitive):Integer;');
15504:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15505:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15506:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
      Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15507:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15508:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15509:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15510:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
```

```
15511:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15512:  Function String_End( Str : String; Cars : Word) : String');
15513:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15514:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15515:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15516:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15517:  Function String_IsNumbers( Str : String) : Boolean');
15518:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15519:  Function StringToCsvCell( aStr : String) : String');
15520: end;
15521:
15522: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15523: begin
15524:  Function LongDayName( aDate : TDate) : String');
15525:  Function LongMonthName( aDate : TDate) : String');
15526:  Function ShortYearOf( aDate : TDate) : byte');
15527:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15528:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15529:  Function SecondsToMinutes( Seconds : Integer) : Double');
15530:  Function MinutesToSeconds( Minutes : Double) : Integer');
15531:  Function MinutesToHours( Minutes : Integer) : Double');
15532:  Function HoursToMinutes( Hours : Double) : Integer');
15533:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15534:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15535:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15536:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15537:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15538:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15539:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15540:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15541:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15542: end;
15543:
15544: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15545: begin
15546:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15547:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15548:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15549:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15550:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15551:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15552:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15553:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15554:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15555:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15556:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15557:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15558:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15559:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15560:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15561:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15562:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15563:  Procedure cyCenterControl( aControl : TControl)');
15564:  Function GetLastParent( aControl : TControl) : TWinControl');
15565:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15566:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15567: end;
15568:
15569: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15570: begin
15571:  Function TablePackTable( Tab : TTable) : Boolean');
15572:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15573:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15574:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15575:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15576:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15577:  Function TableEmptyTable( Tab : TTable) : Boolean');
15578:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15579:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15580:  Function
       TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
15581:  Function
       TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15582:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15583: end;
15584:
15585: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15586: begin
15587:   SIRegister_TcyRunTimeDesign(CL);
15588:   SIRegister_TcyShadowText(CL);
15589:   SIRegister_TcyBgPicture(CL);
15590:   SIRegister_TcyGradient(CL);
15591:   SIRegister_tcyBevel(CL);
15592:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15593:   SIRegister_tcyBevels(CL);
```

```
15594:   SIRegister_TcyImagelistOptions(CL);
15595:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15596: end;
15597:
15598: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15599: begin
15600:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15601:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15602:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15603:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15604:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15605:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15606:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15607:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15608:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
       BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15609:  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;
15610:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15611:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15612:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
       TColor; aState : TButtonState; Focused, Hot : Boolean)');
15613:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
       GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15614:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
       const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15615:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
       TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15616:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
       TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15617:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
       WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15618:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15619:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15620:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15621:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
       CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15622:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15623:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15624:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15625:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15626:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15627:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15628:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15629:  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);
15630:  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;
15631:  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);');
15632:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15633:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15634:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15635:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15636:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15637:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15638:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15639:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15640:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15641:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15642:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15643:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15644:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15645:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15646:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15647:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15648: end;
15649:
15650: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15651: begin
15652:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15653:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15654:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15655:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15656:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
```

```
15657:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15658:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15659:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15660:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15661:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15662:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15663:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
       bmInvertReverseFromColor);
15664:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
       rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15665:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15666:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15667: end;
15668:
15669: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15670: begin
15671:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15672:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15673:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15674:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15675:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15676:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15677:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15678:  Const SC_GROUP_IDENTIFIERW','String') '+');
15679:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15680:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15681:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15682:  Const SERVICE_INACTIVE','LongWord $00000002);
15683:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15684:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15685:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15686:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15687:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15688:  Const SERVICE_STOPPED','LongWord $00000001);
15689:  Const SERVICE_START_PENDING','LongWord $00000002);
15690:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15691:  Const SERVICE_RUNNING','LongWord $00000004);
15692:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15693:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15694:  Const SERVICE_PAUSED','LongWord $00000007);
15695:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15696:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15697:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15698:  Const SC_MANAGER_CONNECT','LongWord $0001);
15699:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15700:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15701:  Const SC_MANAGER_LOCK','LongWord $0008);
15702:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15703:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15704:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15705:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15706:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15707:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15708:  Const SERVICE_START','LongWord $0010);
15709:  Const SERVICE_STOP','LongWord $0020);
15710:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15711:  Const SERVICE_INTERROGATE','LongWord $0080);
15712:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15713:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15714:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15715:  Const SERVICE_ADAPTER','LongWord $00000004);
15716:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15717:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15718:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15719:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15720:  Const SERVICE_BOOT_START','LongWord $00000000);
15721:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15722:  Const SERVICE_AUTO_START','LongWord $00000002);
15723:  Const SERVICE_DEMAND_START','LongWord $00000003);
15724:  Const SERVICE_DISABLED','LongWord $00000004);
15725:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15726:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15727:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15728:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15729:   CL.AddTypeS('SC_HANDLE', 'THandle');
15730:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15731:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15732:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15733:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15734:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15735:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15736:   Const TServiceStatus', '_SERVICE_STATUS');
15737:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15738:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15739:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15740:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15741:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15742:   TEnumServiceStatus', 'TEnumServiceStatusA');
15743:   SC_LOCK', '___Pointer');
```

```
15744:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15745:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15746:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15747:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15748:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15749:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15750:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15751:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15752:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15753:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15754:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15755:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15756:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15757:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15758:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15759:   TQueryServiceConfig', 'TQueryServiceConfigA');
15760:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15761:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15762:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15763:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15764:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15765:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15766:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15767:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
      TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15768:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
      lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
      : DWORD):BOOL');
15769:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
      lpcchBuffer:DWORD):BOOL');
15770:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
      lpcchBuffer:DWORD):BOOL;
15771:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15772:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15773:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15774:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15775:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
      cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15776:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15777:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15778:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15779:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15780: end;
15781:
15782: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15783: begin
15784:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
      AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
      BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15785:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
      DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15786:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15787:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
      TWinControl;
15788:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
      AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15789:  Function CreateNotifyThread(const
      FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15790: end;
15791:
15792: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15793: begin
15794:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15795:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15796:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15797:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15798:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15799:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15800:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15801:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15802:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15803:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15804:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15805:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15806:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15807:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15808:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15809:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15810:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15811:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15812:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15813:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15814:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15815:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15816:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15817:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15818:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
```

```
15819:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15820:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15821:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15822:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15823:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15824:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15825:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15826:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15827:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15828:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15829:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15830:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15831:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15832:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15833:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15834:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15835:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15836:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15837:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15838:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15839:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
      List:TStrings):Bool
15840:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15841:   FindClass('TOBJECT'),'EJclFileSummaryError');
15842:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15843:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15844:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15845:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15846:   SIRegister_TJclFilePropertySet(CL);
15847:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15848:   SIRegister_TJclFileSummary(CL);
15849:   SIRegister_TJclFileSummaryInformation(CL);
15850:   SIRegister_TJclDocSummaryInformation(CL);
15851:   SIRegister_TJclMediaFileSummaryInformation(CL);
15852:   SIRegister_TJclMSISummaryInformation(CL);
15853:   SIRegister_TJclShellSummaryInformation(CL);
15854:   SIRegister_TJclStorageSummaryInformation(CL);
15855:   SIRegister_TJclImageSummaryInformation(CL);
15856:   SIRegister_TJclDisplacedSummaryInformation(CL);
15857:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15858:   SIRegister_TJclMiscSummaryInformation(CL);
15859:   SIRegister_TJclWebViewSummaryInformation(CL);
15860:   SIRegister_TJclMusicSummaryInformation(CL);
15861:   SIRegister_TJclDRMSummaryInformation(CL);
15862:   SIRegister_TJclVideoSummaryInformation(CL);
15863:   SIRegister_TJclAudioSummaryInformation(CL);
15864:   SIRegister_TJclControlPanelSummaryInformation(CL);
15865:   SIRegister_TJclVolumeSummaryInformation(CL);
15866:   SIRegister_TJclShareSummaryInformation(CL);
15867:   SIRegister_TJclLinkSummaryInformation(CL);
15868:   SIRegister_TJclQuerySummaryInformation(CL);
15869:   SIRegister_TJclImageInformation(CL);
15870:   SIRegister_TJclJpegSummaryInformation(CL);
15871: end;
15872:
15873: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15874: begin
15875:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15876:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15877:   T8087Infinity','( icProjective, icAffine )');
15878:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15879:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15880:  Function Get8087ControlWord : Word');
15881:  Function Get8087Infinity : T8087Infinity');
15882:  Function Get8087Precision : T8087Precision');
15883:  Function Get8087Rounding : T8087Rounding');
15884:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15885:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15886:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15887:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15888:  Function Set8087ControlWord( const Control : Word) : Word');
15889:  Function ClearPending8087Exceptions : T8087Exceptions');
15890:  Function GetPending8087Exceptions : T8087Exceptions');
15891:  Function GetMasked8087Exceptions : T8087Exceptions');
15892:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15893:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15894:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15895: end;
15896:
15897: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15898: begin
15899:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15900:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15901:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
15902:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15903:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15904:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15905:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
```

```
15906:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15907: end;
15908:
15909: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15910: begin
15911:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15912:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15913:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15914:   type ULONG', 'Cardinal');
15915:      LPCWSTR', 'PChar');
15916:   CL.AddTypeS('LPWSTR', 'PChar');
15917:   LPSTR', 'PChar');
15918:   TBindVerb', 'ULONG');
15919:   TBindInfoF', 'ULONG');
15920:   TBindF', 'ULONG');
15921:   TBSCF', 'ULONG');
15922:   TBindStatus', 'ULONG');
15923:   TCIPStatus', 'ULONG');
15924:   TBindString', 'ULONG');
15925:   TPiFlags', 'ULONG');
15926:   TOIBdgFlags', 'ULONG');
15927:   TParseAction', 'ULONG');
15928:   TPSUAction', 'ULONG');
15929:   TQueryOption', 'ULONG');
15930:   TPUAF', 'ULONG');
15931:   TSZMFlags', 'ULONG');
15932:   TUrlZone', 'ULONG');
15933:   TUrlTemplate', 'ULONG');
15934:   TZAFlags', 'ULONG');
15935:   TUrlZoneReg', 'ULONG');
15936:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15937:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15938:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15939:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15940:  const 'CF_NULL','LongInt').SetInt( 0);
15941:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15942:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15943:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15944:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15945:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15946:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15947:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15948:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15949:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15950:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15951:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15952:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15953:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15954:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15955:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15956:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15957:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15958:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15959:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15960:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15961:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15962:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15963:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15964:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15965:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15966:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15967:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15968:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15969:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15970:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15971:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15972:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15973:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15974:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15975:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15976:   SIRegister_IPersistMoniker(CL);
15977:   SIRegister_IBindProtocol(CL);
15978:   SIRegister_IBinding(CL);
15979:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15980:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15981:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15982:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15983:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15984:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15985:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15986:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15987:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15988:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15989:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15990:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15991:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15992:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15993:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15994:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
```

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

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

```
16170:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16171:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16172:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16173:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16174:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16175:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16176:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16177:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16178:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16179:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16180:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16181:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16182:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16183:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16184:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16185:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16186:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16187:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16188:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16189:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16190:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16191:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16192:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16193:   SIRegister_IInternetProtocolInfo(CL);
16194:   IOInet', 'IInternet');
16195:   IOInetBindInfo', 'IInternetBindInfo');
16196:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16197:   IOInetProtocol', 'IInternetProtocol');
16198:   IOInetProtocolSink', 'IInternetProtocolSink');
16199:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16200:   IOInetSession', 'IInternetSession');
16201:   IOInetPriority', 'IInternetPriority');
16202:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16203:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16204:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16205:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16206:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16207:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16208:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16209:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16210:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16211:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16212:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16213:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16214:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16215:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16216:   //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16217:  //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16218: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16219: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16220:  //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16221:  //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16222:  //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16223:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16224:   SIRegister_IInternetSecurityMgrSite(CL);
16225:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16226:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16227:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16228:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16229:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16230:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16231:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16232:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16233:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16234:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16235:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16236:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16237:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16238:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16239:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16240:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16241:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16242:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16243:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16244:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16245:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16246:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16247:  const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16248:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16249:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
```

```
16250:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16251:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16252:   SIRegister_IInternetSecurityManager(CL);
16253:   SIRegister_IInternetHostSecurityManager(CL);
16254:   SIRegister_IInternetSecurityManagerEx(CL);
16255:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16256:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16257:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16258:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16259:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16260:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16261:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16262:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16263:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16264:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16265:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16266:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16267:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16268:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16269:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16270:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16271:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16272:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16273:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16274:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16275:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16276:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16277:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16278:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16279:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16280:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16281:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16282:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16283:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16284:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16285:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16286:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16287:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16288:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16289:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16290:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16291:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16292:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16293:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16294:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16295:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16296:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16297:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16298:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16299:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16300:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16301:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16302:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16303:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16304:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16305:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16306:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16307:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16308:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16309:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16310:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16311:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16312:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16313:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16314:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16315:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16316:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16317:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16318:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16319:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16320:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16321:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16322:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16323:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16324:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16325:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16326:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16327:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16328:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16329:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16330:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16331:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16332:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16333:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16334:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16335:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16336:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16337:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16338:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
```

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

```
16426:   SIRegister_IDataFilter(CL);
16427:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16428:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16429:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16430:    +'terFlags : DWORD; end');
16431:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16432:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16433:   //PDataInfo', '^TDataInfo // will not work');
16434:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16435:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16436:   TDataInfo', '_tagDATAINFO');
16437:   DATAINFO', '_tagDATAINFO');
16438:   SIRegister_IEncodingFilterFactory(CL);
16439:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16440:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16441:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16442:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16443:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16444:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16445:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16446:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16447:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16448: end;
16449:
16450: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16451: begin
16452:  Procedure reformatMemo( const m : TCustomMemo)');
16453:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16454:  Procedure MoveToTop( memo : TMemo)');
16455:  Procedure ScrollToTop( memo : TMemo)');
16456:  Function LineNumberClicked( memo : TMemo) : integer');
16457:  Function MemoClickedLine( memo : TMemo) : integer');
16458:  Function ClickedMemoLine( memo : TMemo) : integer');
16459:  Function MemoLineClicked( memo : TMemo) : integer');
16460:  Function LinePositionClicked( Memo : TMemo) : integer');
16461:  Function ClickedMemoPosition( memo : TMemo) : integer');
16462:  Function MemoPositionClicked( memo : TMemo) : integer');
16463:  Procedure AdjustGridSize( grid : TDrawGrid)');
16464:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16465:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16466:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16467:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16468:  Procedure sortstrDown( var s : string)');
16469:  Procedure sortstrUp( var s : string)');
16470:  Procedure rotatestrleft( var s : string)');
16471:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16472:  Function deblank( s : string) : string');
16473:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16474:  Procedure FreeAndClearListBox( C : TListBox);');
16475:  Procedure FreeAndClearMemo( C : TMemo);');
16476:  Procedure FreeAndClearStringList( C : TStringList);');
16477:  Function dffgetfilesize( f : TSearchrec) : int64');
16478: end;
16479:
16480: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16481: begin
16482:   CL.AddTypeS('intset', 'set of byte');
16483:   TPoint64', 'record x : int64; y : int64; end');
16484:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16485:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16486:  Function GeneratePentagon( n : integer) : integer');
16487:  Function IsPentagon( p : integer) : boolean');
16488:  Function isSquare( const N : int64) : boolean');
16489:  Function isCube( const N : int64) : boolean');
16490:  Function isPalindrome( const n : int64) : boolean;');
16491:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16492:  Function GetEulerPhi( n : int64) : int64');
16493:  Function dffIntPower( a, b : int64) : int64;');
16494:  Function IntPower1( a : extended; b : int64) : extended;');
16495:  Function gcd2( a, b : int64) : int64');
16496:  Function GCDMany( A : array of integer) : integer');
16497:  Function LCMMany( A : array of integer) : integer');
16498:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16499:  Function dffFactorial( n : int64) : int64');
16500:  Function digitcount( n : int64) : integer');
16501:  Function nextpermute( var a : array of integer) : boolean');
16502:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16503:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16504:  Function InttoBinaryStr( nn : integer) : string');
16505:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16506:  Function AngleToStr( angle : extended) : string');
16507:  Function deg2rad( deg : extended) : extended');
16508:  Function rad2deg( rad : extended) : extended');
16509:  Function GetLongToMercProjection( const long : extended) : extended');
16510:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16511:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16512:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16513:   SIRegister_TPrimes(CL);
16514:   //RIRegister_TPrimes(CL);
```

```
16515:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16516:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16517:  Function Random64( const N : Int64) : Int64;');
16518:  Procedure Randomize64');
16519:  Function Random641 : extended;');
16520:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16521: end;
16522:
16523: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16524: begin
16525:   TrealPoint', 'record x : extended; y : extended; end');
16526:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16527:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16528:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16529:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16530:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16531:  Function realpoint( x, y : extended) : TRealPoint');
16532:  Function dist( const p1, p2 : TrealPoint) : extended');
16533:  Function intdist( const p1, p2 : TPoint) : integer');
16534:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16535:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16536:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16537:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16538:  Function GetTheta( const L : TLine) : extended;');
16539:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16540:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16541:  Procedure Extendline( var L : TLine; dist : integer);');
16542:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16543:  Function Linesintersect( line1, line2 : TLine) : boolean');
16544:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16545:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16546:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16547:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16548:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16549:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16550:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16551:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16552:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16553:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16554:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16555:  Function DegtoRad( d : extended) : extended');
16556:  Function RadtoDeg( r : extended) : extended');
16557:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16558:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16559:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16560:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16561:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16562:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16563:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16564:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16565:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16566: end;
16567:
16568:
16569: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16570: begin
16571:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16572:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16573:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16574:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16575:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16576:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16577:    +'arth : extended; Phase : extended; end');
16578:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16579:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16580:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16581:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16582:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16583:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16584:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16585:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16586:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16587:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16588:   SIRegister_TAstronomy(CL);
16589:  Function AngleToStr( angle : extended) : string');
16590:  Function StrToAngle( s : string; var angle : extended) : boolean');
16591:  Function HoursToStr24( t : extended) : string');
16592:  Function RPoint( x, y : extended) : TRPoint');
16593:  Function getStimename( t : TDTType) : string');
16594: end;
16595:
16596: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
```

```
16597: begin
16598:   TCardValue', 'Integer');
16599:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16600:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16601:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16602:   SIRegister_TCard(CL);
16603:   SIRegister_TDeck(CL);
16604: end;
16605:
16606: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16607: begin
16608:   tMethodCall', 'Procedure');
16609:   tVerboseCall', 'Procedure ( s : string)');
16610:  // PTEdge', '^TEdge // will not work');
16611:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16612:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16613:   SIRegister_TNode(CL);
16614:   SIRegister_TGraphList(CL);
16615: end;
16616:
16617: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16618: begin
16619:   ParserFloat', 'extended');
16620:   //PParserFloat', '^ParserFloat // will not work');
16621:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16622:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16623:   //POperation', '^TOperation // will not work');
16624:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16625:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16626:    +'; Token : TDFFToken; end');
16627:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16628:   (CL.FindClass('TOBJECT'),'EMathParserError');
16629:   CL.FindClass('TOBJECT'),'ESyntaxError');
16630:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16631:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16632:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16633:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16634:   (CL.FindClass('TOBJECT'),'EBadName');
16635:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16636:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16637:   SIRegister_TCustomParser(CL);
16638:   SIRegister_TExParser(CL);
16639: end;
16640:
16641:   function isService: boolean;
16642:   begin
16643:     result:= NOT(Application is TApplication);
16644:     {result:= Application is TServiceApplication;}
16645:   end;
16646:   function isApplication: boolean;
16647:   begin
16648:     result:= Application is TApplication;
16649:   end;
16650:   //SM_REMOTESESSION = $1000
16651:   function isTerminalSession: boolean;
16652:   begin
16653:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16654:   end;
16655:
16656: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16657: begin
16658:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16659:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16660:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16661:  Function cyURLEncode( const S : string) : string');
16662:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16663:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16664:  Function cyColorToHtml( aColor : TColor) : String');
16665:  Function HtmlToColor( aHtmlColor : String) : TColor');
16666:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16667:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16668:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16669:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16670:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16671:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16672:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16673:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16674:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16675:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16676:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16677:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16678: end;
16679:
16680:
16681: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16682: begin
16683:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16684:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16685:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
```

```
16686:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16687:    +'inationsRepeat, CombinationsRepeatDown )');
16688:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16689:   SIRegister_TComboSet(CL);
16690: end;
16691:
16692:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16693: begin
16694:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16695:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16696:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16697:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16698:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16699:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16700:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16701:   SIRegister_TcyBaseComm(CL);
16702:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16703:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16704:  Function ValidateFileMappingName( aName : String) : String');
16705:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16706: end;
16707:
16708: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16709: begin
16710:   CL.AddTypeS('DERString', 'String');
16711:   CL.AddTypeS('DERChar', 'Char');
16712:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16713:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16714:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16715:   CL.AddTypeS('DERNString', 'String');
16716:  const DERDecimalSeparator','String').SetString( '.');
16717:  const DERDefaultChars','String')('+°@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16718:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16719:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16720:  Function isValidWebMailChar( aChar : Char) : Boolean');
16721:  Function isValidwebSite( aStr : String) : Boolean');
16722:  Function isValidWebMail( aStr : String) : Boolean');
16723:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16724:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16725:  Function IsDERChar( aChar : Char) : Boolean');
16726:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16727:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16728:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16729:  Function IsDERSymbols( aDERString : String) : Boolean');
16730:  Function StringToDERCharSet( aStr : String) : DERString');
16731:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16732:  Function IsDERNChar( aChar : Char) : Boolean');
16733:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16734:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16735:  Function DERExtractWebMail( aDERStr : DERString) : String');
16736:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16737:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16738:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
      RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16739:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
      aElementsType : TElementsType) : String;');
16740:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
      Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16741: end;
16742:
16743: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16744: begin
16745:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16746:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16747:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16748:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16749:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16750:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16751:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16752:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16753:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
      RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16754:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
      PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
      PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16755:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
      SingleDestinationColor, RefreshBmp : Boolean)');
16756:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16757:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16758:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16759:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16760:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16761:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16762:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16763:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16764: end;
```

```
16765:
16766: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16767: begin
16768:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16769:   TPCMChannel', '( cMono, cStereo )');
16770:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16771:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16772:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16773:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16774:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16775:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16776:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16777:    +'it48000Hz, Stereo16bit48000Hz )');
16778:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16779:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16780:    tWaveFormatEx', 'PWaveFormatEx');
16781:    HMMIO', 'Integer');
16782:   TWaveDeviceFormats', 'set of TPCMFormat');
16783:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16784:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16785:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16786:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16787:   TWaveOutOptions', 'set of TWaveOutOption');
16788:   TStreamOwnership2', '( soReference, soOwned )');
16789:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16790:  // PRawWave', '^TRawWave // will not work');
16791:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16792:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16793:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
16794:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16795:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16796:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16797:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16798:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16799:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16800:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16801:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16802:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16803:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16804:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16805:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16806:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16807:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16808:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16809:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16810:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16811:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16812:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16813:  GetAudioFormat( FormatTag : Word) : String');
16814:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16815:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16816:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16817:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16818:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16819:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16820:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16821:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
       TObject; BufferSize : DWORD) : Boolean');
16822:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
       dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16823:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
       TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16824:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16825:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16826:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16827:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16828:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16829:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16830: end;
16831:
16832: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16833: begin
16834:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16835:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16836:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16837:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
16838:  'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16839:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16840:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16841:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16842:   CL.AddTypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg )');
16843:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16844:   SIRegister_TNamedPipe(CL);
16845:   SIRegister_TServerPipe(CL);
16846:   SIRegister_TClientPipe(CL);
16847:  CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD) : DWORD');
16848:  Function HasOverlappedIoCompleted( const ov : OVERLAPPED) : Boolean');
16849:  Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean):
       TOverlappedResult;
```

```
16850:  Function GetStreamAsText( stm : TStream) : string');
16851:  Procedure SetStreamAsText( const aTxt : string; stm : TStream)');
16852: end;
16853:
16854: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16855: begin
16856:  // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16857:   SIRegister_TThumbData(CL);
16858:  'PIC_BMP','LongInt').SetInt( 0);
16859:  'PIC_JPG','LongInt').SetInt( 1);
16860:  'THUMB_WIDTH','LongInt').SetInt( 60);
16861:  'THUMB_HEIGHT','LongInt').SetInt( 60);
16862:  Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean');
16863:  Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)');
16864:  Function ReadBitmap( Filehandle : Integer; Width, Height : Integer) : TBitmap');
16865:  Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap)');
16866:  Function OpenPicture( fn : string; var tp : Integer) : Integer');
16867:  Function ConvertPicture( pi : Integer; tp : Integer) : TBitmap');
16868:  Function LoadPicture( fn : string; var w, h : Integer) : TBitmap');
16869:  Function TurnBitmap( bmp : TBitmap; ang : Integer) : TBitmap');
16870:  Function RotateBitmap( Bitmap : TBitmap; Direction : Integer) : TBitmap');
16871:  Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect');
16872:  Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer) : TBitmap');
16873:  Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer)');
16874:  Procedure FindFiles( path, mask : string; items : TStringList)');
16875:  Function LetFileName( s : string) : string');
16876:  Function LetParentPath( path : string) : string');
16877:  Function AddBackSlash( path : string) : string');
16878:  Function CutBackSlash( path : string) : string');
16879: end;
16880:
16881: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16882: begin
16883:  //'BYTES','LongInt').SetInt( 1);
16884:  'KBYTES','LongInt').SetInt( 1024);
16885:  'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ));
16886:  'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ));
16887:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16888:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
16889:   SIRegister_MakeComServerMethodsPublic(CL);
16890:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16891:  Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) : Boolean');
16892:  Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)');
16893:  Function TBGetTempFolder : string');
16894:  Function TBGetTempFile : string');
16895:  Function TBGetModuleFilename : string');
16896:  Function FormatModuleVersionInfo( const aFilename : string) : string');
16897:  Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD) : string');
16898:  Function TBGetFileSize( aFile : string; aMultipleOf : Integer) : Integer');
16899:  Function FormatAttribString( aAttr : Integer) : string');
16900:  Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo) : string');
16901:  Function ShellRecycle( aWnd : HWND; aFileOrFolder : string) : Boolean');
16902:  Function IsDebuggerPresent : BOOL');
16903:  Function TBNotImplemented : HRESULT');
16904: end;
16905:
16906: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16907: begin
16908:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16909:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16910:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16911:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16912:   //TDrivesProperty = array['A'..'Z'] of boolean;
16913:  Function TBSetSystemTime( DateTime : TDateTime; DOW : word) : boolean');
16914:  Function IsElevated : Boolean');
16915:  Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16916:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16917:  Function TrimNetResource( UNC : string) : string');
16918:  Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16919:  Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16920:  Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
       UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
16921:  Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16922:  Function TBIsWindowsVista : Boolean');
16923:  Procedure SetVistaFonts( const AForm : TForm)');
16924:  Procedure SetVistaContentFonts( const AFont : TFont)');
16925:  Function GetProductType( var sType : String) : Boolean');
16926:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16927:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16928:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16929:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16930:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16931:  Function lstrlen( lpString : PChar) : Integer');
16932:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16933:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16934: end;
16935:
```

```
16936: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16937: begin
16938:  'cLineTerminator','Char').SetString( #10);
16939:  'cLineTerminators','String').SetString( #13#10);
16940:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16941:   SIRegister_TFixedCriticalSection(CL);
16942:   SIRegister_TMultiReadSingleWrite(CL);
16943:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16944:  Function GetDecimalSeparator : Char');
16945:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16946:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16947:   CL.AddTypeS('NativeInt', 'Integer');
16948:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16949:   CL.AddTypeS('NativeUInt', 'Cardinal');
16950:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16951:   //CL.AddTypeS('TBytes', 'array of Byte');
16952:   CL.AddTypeS('RawByteString', 'UnicodeString');
16953:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16954:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16955:   SIRegister_TPath(CL);
16956:   SIRegister_TFile(CL);
16957:   SIRegister_TdwsThread(CL);
16958:  Function GetSystemMilliseconds : Int64');
16959:  Function UTCDateTime : TDateTime');
16960:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16961:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16962:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16963:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16964:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16965:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16966:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16967:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16968:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16969:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16970:  Function InterlockedIncrement( var val : Integer) : Integer');
16971:  Function InterlockedDecrement( var val : Integer) : Integer');
16972:  Procedure FastInterlockedIncrement( var val : Integer)');
16973:  Procedure FastInterlockedDecrement( var val : Integer)');
16974:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16975:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16976:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16977:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16978:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16979:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16980:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16981:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16982:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16983:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16984:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16985:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16986:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
16987:  Procedure CloseFileHandle( hFile : THandle)');
16988:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
16989:  Function FileMove( const existing, new : UnicodeString) : Boolean');
16990:  Function dwsFileDelete( const fileName : String) : Boolean');
16991:  Function FileRename( const oldName, newName : String) : Boolean');
16992:  Function dwsFileSize( const name : String) : Int64');
16993:  Function dwsFileDateTime( const name : String) : TDateTime');
16994:  Function DirectSet8087CW( newValue : Word) : Word');
16995:  Function DirectSetMXCSR( newValue : Word) : Word');
16996:  Function TtoObject( const T: byte) : TObject');
16997:  Function TtoPointer( const T: byte) : ___Pointer');
16998:  Procedure GetMemForT(var T: byte; Size : integer)');
16999:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
17000: end;
17001:
17002: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
17003: begin
17004:  'IPStrSize','LongInt').SetInt( 15);
17005:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17006:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
17007:  'ADWSBASE','LongInt').SetInt( 9000);
17008:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17009:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17010:   SIRegister_EApdSocketException(CL);
17011:   TWsMode', '( wsClient, wsServer )');
17012:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket');
17013:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer)');
17014:   SIRegister_TApdSocket(CL);
17015: end;
17016:
17017: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
17018: begin
17019:   SIRegister_TApdCustomComPort(CL);
17020:   SIRegister_TApdComPort(CL);
17021:  Function ComName( const ComNumber : Word) : ShortString');
17022:  Function SearchComPort( const C : TComponent) : TApdCustomComPort');
17023: end;
```

```
17024:
17025: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
17026: begin
17027:  Function inAddBackslash( const S : String) : String');
17028:  Function PathChangeExt( const Filename, Extension : String) : String');
17029:  Function PathCharCompare( const S1, S2 : PChar) : Boolean');
17030:  Function PathCharIsSlash( const C : Char) : Boolean');
17031:  Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean');
17032:  Function PathCharLength( const S : String; const Index : Integer) : Integer');
17033:  Function inPathCombine( const Dir, Filename : String) : String');
17034:  Function PathCompare( const S1, S2 : String) : Integer');
17035:  Function PathDrivePartLength( const Filename : String) : Integer');
17036:  Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
17037:  Function inPathExpand( const Filename : String) : String');
17038:  Function PathExtensionPos( const Filename : String) : Integer');
17039:  Function PathExtractDir( const Filename : String) : String');
17040:  Function PathExtractDrive( const Filename : String) : String');
17041:  Function PathExtractExt( const Filename : String) : String');
17042:  Function PathExtractName( const Filename : String) : String');
17043:  Function PathExtractPath( const Filename : String) : String');
17044:  Function PathIsRooted( const Filename : String) : Boolean');
17045:  Function PathLastChar( const S : String) : PChar');
17046:  Function PathLastDelimiter( const Delimiters, S : string) : Integer');
17047:  Function PathLowercase( const S : String) : String');
17048:  Function PathNormalizeSlashes( const S : String) : String');
17049:  Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17050:  Function PathPos( Ch : Char; const S : String) : Integer');
17051:  Function PathStartsWith( const S, AStartsWith : String) : Boolean');
17052:  Function PathStrNextChar( const S : PChar) : PChar');
17053:  Function PathStrPrevChar( const Start, Current : PChar) : PChar');
17054:  Function PathStrScan( const S : PChar; const C : Char) : PChar');
17055:  Function inRemoveBackslash( const S : String) : String');
17056:  Function RemoveBackslashUnlessRoot( const S : String) : String');
17057:  Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean)');
17058: end;
17059:
17060:
17061: procedure SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17062: begin
17063:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17064:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17065:  NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17066:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17067:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName');
17068:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString');
17069:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17070:   //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17071:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17072:   SIRegister_TOneShotTimer(CL);
17073:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17074:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17075:  Function NewFileExists( const Name : String) : Boolean');
17076:  Function inDirExists( const Name : String) : Boolean');
17077:  Function FileOrDirExists( const Name : String) : Boolean');
17078:  Function IsDirectoryAndNotReparsePoint( const Name : String) : Boolean');
17079:  Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String');
17080:  Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17081:  Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filename:String): Boolean');
17082:  Function IniKeyExists( const Section, Key, Filename : String) : Boolean');
17083:  Function IsIniSectionEmpty( const Section, Filename : String) : Boolean');
17084:  Function SetIniString( const Section, Key, Value, Filename : String) : Boolean');
17085:  Function SetIniInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean');
17086:  Function SetIniBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean');
17087:  Procedure DeleteIniEntry( const Section, Key, Filename : String)');
17088:  Procedure DeleteIniSection( const Section, Filename : String)');
17089:  Function GetEnv( const EnvVar : String) : String');
17090:  Function GetCmdTail : String');
17091:  Function GetCmdTailEx( StartIndex : Integer) : String');
17092:  Function NewParamCount : Integer');
17093:  Function NewParamStr( Index : Integer) : string');
17094:  Function AddQuotes( const S : String) : String');
17095:  Function RemoveQuotes( const S : String) : String');
17096:  Function inGetShortName( const LongName : String) : String');
17097:  Function inGetWinDir : String');
17098:  Function inGetSystemDir : String');
17099:  Function GetSysWow64Dir : String');
17100:  Function GetSysNativeDir( const IsWin64 : Boolean) : String');
17101:  Function inGetTempDir : String');
17102:  Function StringChange( var S : String; const FromStr, ToStr : String) : Integer');
17103:  Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17104:  Function AdjustLength( var S : String; const Res : Cardinal) : Boolean');
17105:  Function UsingWinNT : Boolean');
17106:  Function ConvertConstPercentStr( var S : String) : Boolean');
17107:  Function ConvertPercentStr( var S : String) : Boolean');
17108:  Function ConstPos( const Ch : Char; const S : String) : Integer');
17109:  Function SkipPastConst( const S : String; const Start : Integer) : Integer');
17110:  Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String) : Boolean');
17111:  Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String):Boolean;
17112:  Function RegValueExists( H : HKEY; Name : PChar) : Boolean');
```

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

```
17191:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17192:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
        TWin32FindData) : THandle');
17193:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17194:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17195:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
        TFileVersionNumbers) : Boolean');
17196:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17197:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17198:  Function MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilen,NewFilename:String;const
        Flags:DWORD):BOOL;
17199:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17200:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17201:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17202:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17203:   SIRegister_TFileRedir(CL);
17204:   SIRegister_TTextFileReaderRedir(CL);
17205:   SIRegister_TTextFileWriterRedir(CL);
17206: end;
17207:
17208: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17209: begin
17210:   //CL.AddTypeS('LongWord', 'Cardinal');
17211:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17212:  Function Compare64( const N1, N2 : Integer64) : Integer');
17213:  Procedure Dec64( var X : Integer64; N : LongWord)');
17214:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17215:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17216:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
17217:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17218:  Function Integer64ToStr( X : Integer64) : String');
17219:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17220:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17221:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17222:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17223:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17224: end;
17225:
17226: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17227: begin
17228:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17229:   SIRegister_TSimpleStringList(CL);
17230:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17231:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17232:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17233:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17234:  // TMD5Digest = array[0..15] of Byte;
17235:   //     TSHA1Digest = array[0..19] of Byte;
17236:  Function CheckForMutexes( Mutexes : String) : Boolean');
17237:  Function CreateTempDir : String');
17238:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17239:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
        FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17240:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
        DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
        TDeleteFileProc; const Param : Pointer) : Boolean');
17241:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
        TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
        TDetermineDefaultLanguageResult');
17242:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17243:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17244:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17245:  Function GetComputerNameString : String');
17246:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17247:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17248:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17249: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17250:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17251:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17252: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17253:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17254:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
        TotBytes:Int64):Bool;
17255:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
        TotalBytes: Integer64):Bool;
17256:  Function GetUserNameString : String');
17257:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17258:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
        String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
        ResultCode:Integer) : Boolean');
17259: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
        TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17260:  Procedure InternalError( const Id : String)');
17261:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17262:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17263:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17264:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17265:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17266:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
```

```
17267:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17268:  Procedure RefreshEnvironment');
17269:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17270:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17271:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17272:  Function RestartComputer : Boolean');
17273:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17274:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17275:  Procedure Win32ErrorMsg( const FunctionName : String)');
17276:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17277:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17278:  //from Func2
17279:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
       IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
17280:   //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17281:  Procedure RegisterTypeLibrary( const Filename : String)');
17282:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17283:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17284:  function getVersionInfoEx3: TOSVersionInfoEx;');
17285:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17286:  procedure InitOle;');
17287:  Function ExpandConst( const S : String) : String');
17288:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17289:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17290:  Function ExpandConstIfPrefixed( const S : String) : String');
17291:  Procedure LogWindowsVersion');
17292:  Function EvalCheck( const Expression : String) : Boolean');
17293: end;
17294:
17295: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17296: begin
17297:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17298:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17299:   SIRegister_TResourceModule(CL);
17300:   SIRegister_TResourceDetails(CL);
17301:   SIRegister_TAnsiResourceDetails(CL);
17302:   SIRegister_TUnicodeResourceDetails(CL);
17303:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17304:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17305:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17306:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17307:  Function ResourceNameToInt( const s : string) : Integer');
17308:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17309: end;
17310:
17311:
17312: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17313: begin
17314:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17315:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
17316:     RegisterMethod('Constructor Create');
17317:     RegisterMethod('Procedure Free');
17318:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17319:     RegisterMethod('Procedure WriteString( const S : String)');
17320:     RegisterMethod('Procedure ReadString( var S : String)');
17321:   end;
17322:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17323:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17324:       SimpleComPort.WriteString(AsciiChar);
17325: end;
17326:
17327:
17328: procedure SIRegister_Console(CL: TPSPascalCompiler);
17329: begin
17330:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17331: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17332:  ('conBW40','LongInt').SetInt( 0);
17333:  ('conCO40','LongInt').SetInt( 1);
17334:  ('conBW80','LongInt').SetInt( 2);
17335:  ('conCO80','LongInt').SetInt( 3);
17336:  ('conMono','LongInt').SetInt( 7);
17337:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17338:  //CL.AddConstantN('C40','').SetString( CO40);
17339:  //CL.AddConstantN('C80','').SetString( CO80);
17340:  Function conReadKey : Char');
17341:  Function conKeyPressed : Boolean');
17342:  Procedure conGotoXY( X, Y : Smallint)');
17343:  Function conWhereX : Integer');
17344:  Function conWhereY : Integer');
17345:  Procedure conTextColor( Color : Byte);');
17346:  Function conTextColor1 : Byte;');
17347:  Procedure conTextBackground( Color : Byte);');
17348:  Function conTextBackground1 : Byte;');
17349:  Procedure conTextMode( Mode : Word)');
17350:  Procedure conLowVideo');
17351:  Procedure conHighVideo');
17352:  Procedure conNormVideo');
17353:  Procedure conClrScr');
17354:  Procedure conClrEol');
```

```
17355:  Procedure conInsLine');
17356:  Procedure conDelLine');
17357:  Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17358:  Function conScreenWidth : Smallint');
17359:  Function conScreenHeight : Smallint');
17360:  Function conBufferWidth : Smallint');
17361:  Function conBufferHeight : Smallint');
17362:  procedure InitScreenMode;');
17363: end;
17364:
17365: (*----------------------------------------------------------------------------*)
17366: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17367: begin
17368:   SIRegister_TNoRefCountObject(CL);
17369:  Procedure FreeObjects( List : TFPList)');
17370:  Procedure GetMethodList( AObject : TObject; AList : TStrings);');
17371:  Procedure GetMethodList1( AClass : TClass; AList : TStrings);');
17372: end;
17373:
17374: procedure SIRegister_ToolsUnit(CL: TPSPascalCompiler);
17375: begin
17376:  'MaxDataSet','LongInt').SetInt( 35);
17377:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17378:   SIRegister_TDBConnector(CL);
17379:   SIRegister_TDBBasicsTestSetup(CL);
17380:   SIRegister_TTestDataLink(CL);
17381:  'testValuesCount','LongInt').SetInt( 25);
17382:  Procedure InitialiseDBConnector');
17383:  Procedure FreeDBConnector');
17384:  Function DateTimeToTimeString( d : tdatetime) : string');
17385:  Function TimeStringToDateTime( d : String) : TDateTime');
17386: end;
17387:
17388: procedure SIRegister_fpcunit(CL: TPSPascalCompiler);
17389: begin
17390:   SIRegister_EAssertionFailedError(CL);
17391:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17392:   CL.AddTypeS('TRunMethod', 'Procedure');
17393:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTestResult');
17394:   SIRegister_TTest(CL);
17395:   SIRegister_TAssert(CL);
17396:   SIRegister_TTestFailure(CL);
17397:   SIRegister_ITestListener(CL);
17398:   SIRegister_TTestCase(CL);
17399:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17400:   SIRegister_TTestSuite(CL);
17401:   SIRegister_TTestResult(CL);
17402:  Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17403: end;
17404:
17405: procedure SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
17406: begin
17407:   TOBJECT'),'ETCPBuffer');
17408:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Intege'
17409:    +r; Head : Integer; Used : Integer; end');
17410:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17411:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17412:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
17413:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
17414:  Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int;
17415:  Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17416:  Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17417:  Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17418:  Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17419:  Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17420:  Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17421:  Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17422:  Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17423:  Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17424:  Function TCPBufferRemove(var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17425:  Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17426:  Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17427:  Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');
17428:  Function TCPBufferEmpty( const TCPBuf : TTCPBuffer) : Boolean');
17429:  Function TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Integer');
17430:  Function TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer');
17431:  Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer)');
17432: end;
17433:
17434: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17435: begin
17436:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17437:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17438:  CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
17439:  CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
17440:  CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0);
17441:  CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
17442:  CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
17443:  Procedure LoadGlut( const dll : String)');
```

```
17444:  Procedure FreeGlut');
17445: end;
17446:
17447: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17448: begin
17449:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17450:  Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean) : THandle');
17451: end;
17452:
17453: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17454: begin
17455:   TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )');
17456:   TTLedState', '( LedOn, LedOff, LedDisabled )');
17457:   SIRegister_TSwitchLed(CL);
17458:  //CL.AddDelphiFunction('Procedure Register');
17459: end;
17460:
17461: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17462: begin
17463:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17464:    +'xisting, fdOpenAlways, fdTruncateExisting )');
17465:   CL.AddTypeS('TIFileAccess', '( faRead, faWrite, faReadWrite )');
17466:   CL.AddTypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17467:   SIRegister_TCustomFile(CL);
17468:   SIRegister_TIFile(CL);
17469:   SIRegister_TMemoryFile(CL);
17470:   SIRegister_TTextFileReader(CL);
17471:   SIRegister_TTextFileWriter(CL);
17472:   SIRegister_TFileMapping(CL);
17473:   SIRegister_EFileError(CL);
17474: end;
17475:
17476: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17477: begin
17478:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17479:    +', ffaDirectory, ffaArchive, ffaAnyFile )');
17480:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17481:   SIRegister_TFileSearch(CL);
17482: end;
17483:
17484: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17485: begin
17486:  TRGBType', 'record RedHex : string; GreenHex : string; BlueHex :'
17487:    + string; Red : integer; Green : integer; Blue : integer; end');
17488:  Function FadeColor( aColor : Longint; aFade : integer) : Tcolor');
17489:  Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17490: end;
17491:
17492: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17493: begin
17494:  Procedure SaveOscSettings');
17495:  Procedure GetOscSettings');
17496: end;
17497:
17498: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17499: begin
17500:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer)');
17501:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17502:    FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17503:    64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17504:    Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17505:   SIRegister_TcyDebug(CL);
17506: end;
17507:
17508: (*----------------------------------------------------------------------------*)
17509: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17510: begin
17511:   TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )');
17512:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )');
17513:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )');
17514:   SIRegister_TDestinationOptions(CL);
17515:   TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult)');
17516:   TProcOnCopyFileProgressEvent','Procedure(Sender:TObject;FileBytes,
      TransferedBytes:int64;PercentDone:Int64);
17517:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17518:   SIRegister_TcyCopyFiles(CL);
17519:  Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
      ResetAttr:boolean): TCopyFileResult');
17520:  Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
      TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17521:  Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
       ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;'
17522:                       + FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17523: end;
17524:
17525: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17526: begin
17527:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17528:   SIRegister_TcyFileAttributes(CL);
```

```
17529:   SIRegister_TSearchRecInstance(CL);
17530:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17531:   TOptions', 'set of TOption');
17532:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17533:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
       Accept:bool;
17534:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17535:   SIRegister_TcyCustomSearchFiles(CL);
17536:   SIRegister_TcySearchFiles(CL);
17537:  Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17538:  Function IscyFolder( aSRec : TSearchrec) : Boolean');
17539: end;
17540:
17541: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17542: begin
17543:  Function jCountChar( const s : string; ch : char) : integer');
17544:  Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings)');
17545:  Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17546:  Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17547:  Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17548:  Function jNormalizeDateSeparator( const s : string) : string');
17549:  Function jIsValidDateString( const Value : string) : boolean');
17550:  Function jIsValidTimeString( const Value : string) : boolean');
17551:  Function jIsValidDateTimeString( const Value : string) : boolean');
17552: end;
17553:
17554: procedure SIRegister_kcMapViewer(CL: TPSPascalCompiler);
17555: begin
17556:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapViewer');
17557:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
17558:    +'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
17559:    +', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
17560:    +'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
17561:    +'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17562:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17563:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17564:   TIntPoint', 'record X : Int64; Y : Int64; end');
17565:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17566:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17567:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17568:   SIRegister_TCustomDownloadEngine(CL);
17569:   SIRegister_TCustomGeolocationEngine(CL);
17570:   SIRegister_TMapViewer(CL);
17571: end;
17572:
17573: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17574: begin
17575:  (*CL.AddDelphiFunction('Function isFunc( name : TNamePart) : Boolean');*)
17576:  CL.AddDelphiFunction('Function isUnnamedFunc( name : TNamepart) : Boolean');
17577:  Function isPtrToFunc( name : TNamePart) : Boolean');
17578:  Function isFuncRetFuncPtr( name : TNamePart) : Boolean');
17579:  Function isPtrToFuncRetFuncPtr( name : TNamePart) : Boolean');
17580:  Function GetFuncParam( name : TNamePart) : TNamePart');
17581:  Function isArray( name : TNamePart) : Boolean');
17582:  Function GetArrayPart( name : TNamePart) : TNamePart');
17583:  Function GetIdFromPart( name : TNamePart) : AnsiString');
17584:  Function GetIdPart( name : TNamePart) : TNamePart');
17585:  Function isNamePartPtrToFunc( part : TNamePart) : Boolean');
17586:  Function isAnyBlock( part : TNamePart) : Boolean');*)
17587:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17588:   SIRegister_TLineBreaker(CL);
17589:   CL.AddTypeS('TNameKind', 'Integer');
17590:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17591:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17592:  Function SphericalMod( X : Extended) : Extended');
17593:  Function cSign( Value : Extended) : Extended');
17594:  Function LimitFloat( const eValue, eMin, eMax : Extended) : Extended');
17595:  Function AngleToRadians( iAngle : Extended) : Extended');
17596:  Function RadiansToAngle( eRad : Extended) : Extended');
17597:  Function Cross180( iLong : Double) : Boolean');
17598:  Function Mod180( Value : integer) : Integer');
17599:  Function Mod180Float( Value : Extended) : Extended');
17600:  Function MulDivFloat( a, b, d : Extended) : Extended');
17601:  Function LongDiff( iLong1, iLong2 : Double) : Double');
17602:  Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
17603:  Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17604:  Function FixFilePath( const Inpath, CheckPath : string) : string');
17605:  Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17606:  Procedure FillStringList( sl : TStringList; const aText : string)');
17607: end;
17608:
17609: procedure SIRegister_LedNumber(CL: TPSPascalCompiler);
17610: begin
17611:   TLedSegmentSize', 'Integer');
17612:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
17613:   SIRegister_TCustomLEDNumber(CL);
17614:   SIRegister_TLEDNumber(CL);
17615: end;
17616:
```

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

```
17702:  Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
        SubString:AnsiString):Bool;
17703:  Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean) : AnsiString');
17704:  Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
        var SubString : AnsiString) : Boolean');
17705:  Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
        SubString : AnsiString) : Boolean');
17706:  Function DeleteWithinL( const S, Delimiter : AnsiString) : AnsiString');
17707:  Function ExtractTokensL(const S,
        Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17708:  Function IsChAlphaL( C : AnsiChar) : Boolean');
17709:  Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17710:  Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17711:  Function IsStrAlphaL( const S : AnsiString) : Boolean');
17712:  Function IsStrNumericL( const S, Numbers : AnsiString) : Boolean');
17713:  Function IsStrAlphaNumericL( const S, Numbers : AnsiString) : Boolean');
17714:  Function KeepCharsL( const S, Chars : AnsiString) : AnsiString');
17715:  Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal) : Boolean');
17716:  Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal) : Boolean');
17717:  Function LastStringL( const S, AString : AnsiString; var Position : Cardinal) : Boolean');
17718:  Function LeftTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17719:  Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Card;var
        Replacements:Card):AnsiString;
17720:  Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
        Replacements:Cardinal):AnsiString');
17721:  Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
        Replacements:Cardinal):AnsiString;
17722:  Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17723:  Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17724:  Function RightTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17725:  Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal) : boolean');
17726:  Function TrimCharsL( const S, Chars : AnsiString) : AnsiString');
17727:  Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17728:  Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17729:  end;
17730:
17731:  procedure SIRegister_pwnative_out(CL: TPSPascalCompiler);
17732: begin
17733:  CL.AddConstantN('STDIN','LongInt').SetInt( 0);
17734:  ('STDOUT','LongInt').SetInt( 1);
17735:  ('STDERR','LongInt').SetInt( 2);
17736:  Procedure NativeWrite( s : astr);');
17737:  Procedure NativeWrite1( PString : PChar);');
17738:  Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal);');
17739:  Procedure NativeWriteLn( s : astr);');
17740:  Procedure NativeWriteLn1;');
17741: end;
17742:
17743: procedure SIRegister_synwrap1(CL: TPSPascalCompiler);
17744: begin
17745:   CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse
17746:     : integer; UltimateURL : ansistring; Headers : ansistring; end');
17747:   CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17748:  Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean): TSynwInfo;');
17749:  Function GetHttpFile1( const Url, UserAgent, Outfile : string) : TSynwInfo;');
17750:  Function GetHttpFile2( const Url, Outfile : string) : TSynwInfo;');
17751:  Function GetHttpFile3( const Url, outfile : string; verbose : boolean) : TSynwInfo;');
17752:  Function GetHtm( const Url : string) : string;');
17753:  Function GetHtm1( const Url, UserAgent : string) : string;');
17754:  Function GetUrl( const Url : string; verbose : boolean) : TSynwInfo;');
17755:  Function GetUrl1( const Url, useragent : string) : TSynwInfo;');
17756:  Function GetUrl2( const Url : string) : TSynwInfo;');
17757:  Function GetUrl3( const Url : string; const http : THTTPSend; verbose : boolean): TUrlInfo;');
17758:  Function GetUrl4( const Url : string; const http : THTTPSend) : TUrlInfo;');
17759:  Procedure StrToStream( s : String; strm : TMemoryStream)');
17760:  Function StrLoadStream( strm : TStream) : String');
17761:
17762: end;
17763:
17764: procedure SIRegister_HTMLUtil(CL: TPSPascalCompiler);
17765: begin
17766:  Function GetVal( const tag, attribname_ci : string) : string');
17767:  Function GetTagName( const Tag : string) : string');
17768:  Function GetUpTagName( const tag : string) : string');
17769:  Function GetNameValPair( const tag, attribname_ci : string) : string');
17770:  Function GetValFromNameVal( const namevalpair : string) : string');
17771:  Function GetNameValPair_cs( const tag, attribname : string) : string');
17772:  Function GetVal_JAMES( const tag, attribname_ci : string) : string');
17773:  Function GetNameValPair_JAMES( const tag, attribname_ci : string) : string');
17774:  Function CopyBuffer( StartIndex : PChar; Len : integer) : string');
17775:  Function Ucase( s : string) : string');
17776: end;
17777:
17778: procedure SIRegister_pwmain(CL: TPSPascalCompiler);
17779: begin
17780:  CL.AddConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT');
17781:  EXPIRED_COOKIE','String').SetString( 'Mon, 01 Jan 2001 12:00:00 GMT');
17782:  'SECURE_OFF','LongInt').SetInt( 0);
17783:  'SECURE_ON','LongInt').SetInt( 2);
```

```
17784:  'SECURE_FILTER','LongInt').SetInt( 3);
17785:   THandle or DWord!
17786:  //  astr = ansistring;
17787:   CL.AddTypeS('pastr', 'ansistring');
17788:   CL.AddTypeS('TFilterFunc', 'function(const s: pastr): pastr;');
17789:   uses pwinit at begin
17790:   //type  TFilterFunc = function(const s: astr): astr;
17791:  Demo: ..\maxbox3\examples2\519_powtils.txt
17792:
17793:  //CL.AddConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false);
17794:  //CL.AddConstantN('CASE_IGNORE','Boolean')BoolToStr( false);
17795:  Procedure pwInit');
17796:  Procedure OffReadln');
17797:  Function Lcase( const s : pastr) : pastr');
17798:  Function Ucase( const s : pastr) : pastr');
17799:  Function CountPostVars : longword');
17800:  Function GetPostVar( const name : pastr) : pastr;');
17801:  Function GetPostVar1( const name : pastr; filter : TFilterFunc) : pastr;');
17802:  Function GetPostVar_S( const name : pastr; Security : integer) : pastr');
17803:  Function GetPostVar_SF( const name : pastr; Security : integer) : pastr');
17804:  Function GetPostVarAsFloat( const name : pastr) : double');
17805:  Function GetPostVarAsInt( const name : pastr) : longint');
17806:  Function GetPostVar_SafeHTML( const name : pastr) : pastr');
17807:  Function FetchPostVarName( idx : longword) : pastr');
17808:  Function FetchPostVarVal( idx : longword) : pastr;');
17809:  Function FetchPostVarVal1( idx : longword; filter : TFilterFunc) : pastr;');
17810:  Function FetchPostVarName_S( idx : longword; Security : integer) : pastr');
17811:  Function FetchPostVarVal_S( idx : longword; Security : integer) : pastr');
17812:  Function IsPostVar( const name : pastr) : boolean');
17813:  Function CountAny : longword');
17814:  Function GetAny( const name : pastr) : pastr;');
17815:  Function GetAny1( const name : pastr; filter : TFilterFunc) : pastr;');
17816:  Function GetAny_S( const name : pastr; Security : integer) : pastr');
17817:  Function GetAnyAsFloat( const name : pastr) : double');
17818:  Function GetAnyAsInt( const name : pastr) : longint');
17819:  Function IsAny( const name : pastr) : byte');
17820:  Function CountCookies : longword');
17821:  Function FetchCookieName( idx : longword) : pastr');
17822:  Function FetchCookieVal( idx : longword) : pastr;');
17823:  Function FetchCookieVal1( idx : longword; filter : TFilterFunc) : pastr;');
17824:  Function GetCookie( const name : pastr) : pastr;');
17825:  Function GetCookie1( const name : pastr; filter : TFilterFunc) : pastr;');
17826:  Function GetCookieAsFloat( const name : pastr) : double');
17827:  Function GetCookieAsInt( const name : pastr) : longint');
17828:  Function IsCookie( const name : pastr) : boolean');
17829:  Function SetCookie( const name, value : pastr) : boolean');
17830:  Function SetCookieAsFloat( const name : pastr; value : double) : boolean');
17831:  Function SetCookieAsInt( const name : pastr; value : longint) : boolean');
17832:  Function SetCookieEx( const name, value, path, domain, expiry : pastr) : boolean');
17833:  Function SetCookieAsFloatEx(const name:pastr;value : double; const path,domain,expiry:pastr):bool;
17834:  Function SetCookieAsIntEx( const name:pastr;value : longint; const path,domain,expiry:pastr):bool;
17835:  Function UnsetCookie( const name : pastr) : boolean');
17836:  Function UnsetCookieEx( const name, path, domain : pastr) : boolean');
17837:  Function FilterHtml( const input : pastr) : pastr');
17838:  Function FilterHtml_S( const input : pastr; security : integer) : pastr');
17839:  Function TrimBadChars( const input : pastr) : pastr');
17840:  Function TrimBadFile( const input : pastr) : pastr');
17841:  Function TrimBadDir( const input : pastr) : pastr');
17842:  Function TrimBad_S( const input : pastr; security : integer) : pastr');
17843:  Function CountHeaders : longword');
17844:  Function FetchHeaderName( idx : longword) : pastr');
17845:  Function FetchHeaderVal( idx : longword) : pastr');
17846:  Function GetHeader( const name : pastr) : pastr');
17847:  Function IsHeader( const name : pastr) : boolean');
17848:  Function SetHeader( const name, value : pastr) : boolean');
17849:  Function UnsetHeader( const name : pastr) : boolean');
17850:  Function PutHeader( const header : pastr) : boolean');
17851:  Procedure Out1( const s : pastr)');
17852:  Procedure OutLn( const s : pastr)');
17853:  Procedure OutA( args : array of const)');
17854:  Procedure OutF( const s : pastr)');
17855:  Procedure OutLnF( const s : pastr)');
17856:  Procedure OutFF( const s : pastr)');
17857:  Procedure OutF_FI( const s : pastr; HTMLFilter : boolean)');
17858:  Procedure OutLnFF( const s : pastr)');
17859:  Procedure OutLnF_FI( const s : pastr; HTMLFilter : boolean)');
17860:  Function FileOut( const fname : pastr) : word');
17861:  Function ResourceOut( const fname : pastr) : word');
17862:  Procedure BufferOut( const buff, len : LongWord)');
17863:  Function TemplateOut( const fname : pastr; HtmlFilter : boolean) : word;');
17864:  Function TemplateOut1( const fname : pastr) : word;');
17865:  Function TemplateOut2( const fname : pastr; filter : TFilterFunc) : word;');
17866:  Function TemplateOut3( const fname : pastr; HtmlFilter : boolean) : word;');
17867:  Function TemplateRaw( const fname : pastr) : word');
17868:  Function Fmt( const s : pastr) : pastr;');
17869:  Function Fmt1( const s : pastr; filter : TFilterFunc) : pastr;');
17870:  Function FmtFilter( const s : pastr) : pastr');
17871:  Function Fmt_SF(const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecurity,TrimSecurity:int):pastr;
17872:  Function Fmt_SF1(const s:pastr; HTMLFilter:boolean; FilterSecurity, TrimSecurity : integer) : pastr;');
```

```
17873:  Function CountRtiVars : longword');
17874:  Function FetchRtiName( idx : longword) : pastr');
17875:  Function FetchRtiVal( idx : longword) : pastr');
17876:  Function GetRti( const name : pastr) : pastr');
17877:  Function GetRtiAsFloat( const name : pastr) : double');
17878:  Function GetRtiAsInt( const name : pastr) : longint');
17879:  Function IsRti( const name : pastr) : boolean');
17880:  Procedure SetRTI( const name, value : pastr)');
17881:  Function FetchUpfileName( idx : longword) : pastr');
17882:  Function GetUpfileName( const name : pastr) : pastr');
17883:  Function GetUpfileSize( const name : pastr) : longint');
17884:  Function GetUpfileType( const name : pastr) : pastr');
17885:  Function CountUpfiles : longword');
17886:  Function IsUpfile( const name : pastr) : boolean');
17887:  Function SaveUpfile( const name, fname : pastr) : boolean');
17888:  Function CountVars : longword');
17889:  Function FetchVarName( idx : longword) : pastr');
17890:  Function FetchVarVal( idx : longword) : pastr;');
17891:  Function FetchVarVal1( idx : longword; filter : TFilterFunc) : pastr;');
17892:  Function GetVar( const name : pastr) : pastr;');
17893:  Function GetVar1( const name : pastr; filter : TFilterFunc) : pastr;');
17894:  Function GetVar_S( const name : pastr; security : integer) : pastr');
17895:  Function GetVarAsFloat( const name : pastr) : double');
17896:  Function GetVarAsInt( const name : pastr) : longint');
17897:  Procedure SetVar( const name, value : pastr)');
17898:  Procedure SetVarAsFloat( const name : pastr; value : double)');
17899:  Procedure SetVarAsInt( const name : pastr; value : longint)');
17900:  Function IsVar( const name : pastr) : byte');
17901:  Procedure UnsetVar( const name : pastr)');
17902:  Function LineEndToBR( const s : pastr) : pastr');
17903:  Function RandomStr( len : longint) : pastr');
17904:  Function XorCrypt( const s : pastr; key : byte) : pastr');
17905:  Function CountCfgVars : longword');
17906:  Function FetchCfgVarName( idx : longword) : pastr');
17907:  Function FetchCfgVarVal( idx : longword) : pastr');
17908:  Function IsCfgVar( const name : pastr) : boolean');
17909:  Function SetCfgVar( const name, value : pastr) : boolean');
17910:  Function GetCfgVar( const name : pastr) : pastr');
17911:  Procedure ThrowErr( const s : pastr)');
17912:  Procedure ThrowWarn( const s : pastr)');
17913:  Procedure ErrWithHeader( const s : pastr)');
17914:   CL.AddTypeS('TWebVar', 'record name : pastr; value : pastr; end');
17915:   CL.AddTypeS('TWebVars', 'array of TWebVar');
17916:  Function iUpdateWebVar(var webv : TWebVars; const name, value:pastr; upcased:boolean):boolean');
17917:  Function iAddWebCfgVar( const name, value : pastr) : boolean');
17918:  Procedure iAddWebVar( var webv : TWebVars; const name, value : pastr)');
17919:  Procedure iSetRTI( const name, value : pastr)');
17920:  Function iCustomSessUnitSet : boolean');
17921:  Function iCustomCfgUnitSet : boolean');
17922: end;
17923:
17924: function TRestRequest_createStringStreamFromStringList(strings: TStringList): TStringStream;
17925:
17926: {A simple Oscilloscope using  TWaveIn class.
17927:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
17928: uses
17929:   Forms,
17930:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
17931:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
17932:   uColorFunctions in 'uColorFunctions.pas',
17933:   AMixer in 'AMixer.pas',
17934:   uSettings in 'uSettings.pas',
17935:   UWavein4 in 'UWavein4.pas',
17936:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
17937:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
17938:
17939:
17940: Functions_max hex in the box maXbox
17941: functionslist.txt
17942: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98/100
17943:
17944: ********************************************************************************
17945: Procedure
17946: PROCEDURE SIZE 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
17947: Procedure ***************************Now the Procedure list******************
17948: Procedure ( ACol, ARow : Integer; Items : TStrings)
17949: Procedure ( Agg : TAggregate)
17950: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
17951: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
17952: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
17953: Procedure ( ASender : TObject; const ABytes : Integer)
17954: Procedure ( ASender : TObject; VStream : TStream)
17955: Procedure ( AThread : TIdThread)
17956: Procedure ( AWebModule : TComponent)
17957: Procedure ( Column : TColumn)
17958: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
17959: Procedure ( const iStart : integer; const sText : string)
17960: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
17961: Procedure ( Database : TDatabase; LoginParams : TStrings)
```

```
17962: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
17963: Procedure ( DATASET : TDATASET)
17964: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
17965: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
17966: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
17967: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
17968: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
17969: Procedure ( Done : Integer)
17970: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
17971: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
17972: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
17973: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
17974: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
17975: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
17976: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
17977: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
17978: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
17979: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
17980: Procedure ( SENDER : TFIELD; const TEXT : String)
17981: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
17982: Procedure ( Sender : TIdTelnet; const Buffer : String)
17983: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
17984: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
17985: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17986: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
17987: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
17988: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
17989: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
17990: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
17991: Procedure ( Sender : TObject; Button : TMPBtnType)
17992: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
17993: Procedure ( Sender : TObject; Button : TUDBtnType)
17994: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
17995: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
17996: Procedure ( Sender : TObject; Column : TListColumn)
17997: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
17998: Procedure ( Sender : TObject; Connecting : Boolean)
17999: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
18000: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
18001: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
18002: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
18003: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
18004: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
18005: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
18006: Procedure ( Sender : TObject; Index : LongInt)
18007: Procedure ( Sender : TObject; Item : TListItem)
18008: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
18009: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
18010: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
18011: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
18012: Procedure ( Sender : TObject; Item : TListItem; var S : string)
18013: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
18014: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
18015: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
18016: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
18017: Procedure ( Sender : TObject; Node : TTreeNode)
18018: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
18019: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
18020: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
18021: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
18022: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
18023: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
18024: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
18025: Procedure ( Sender : TObject; Rect : TRect)
18026: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
18027: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
18028: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
18029: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
18030: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
18031: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
18032: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
18033: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
18034: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
18035: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
18036: Procedure ( Sender : TObject; Thread : TServerClientThread)
18037: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
18038: Procedure ( Sender : TObject; Username, Password : string)
18039: Procedure ( Sender : TObject; var AllowChange : Boolean)
18040: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
18041: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
18042: Procedure ( Sender : TObject; var Continue : Boolean)
18043: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
18044: Procedure ( Sender : TObject; var Username : string)
18045: Procedure ( Sender : TObject; Wnd : HWND)
18046: Procedure ( Sender : TToolbar; Button : TToolButton)
18047: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
18048: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
```

```
18049: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
18050: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
18051: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
18052: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
18053: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
18054: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
18055: procedure (Sender: TObject)
18056: procedure (Sender: TObject; var Done: Boolean)
18057: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
18058: procedure _T(Name: tbtString; v: Variant);
18059: Procedure AbandonSignalHandler( RtlSigNum : Integer)
18060: Procedure Abort
18061: Procedure About1Click( Sender : TObject)
18062: Procedure Accept( Socket : TSocket)
18063: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
18064: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
18065: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
18066: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
18067: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
18068: Procedure Add( Addend1, Addend2 : TMyBigInt)
18069: Procedure ADD( const AKEY, AVALUE : VARIANT)
18070: Procedure Add( const Key : string; Value : Integer)
18071: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
18072: Procedure ADD( FIELD : TFIELD)
18073: Procedure ADD( ITEM : TMENUITEM)
18074: Procedure ADD( POPUP : TPOPUPMENU)
18075: Procedure AddCharacters( xCharacters : TCharSet)
18076: Procedure AddDriver( const Name : string; List : TStrings)
18077: Procedure AddImages( Value : TCustomImageList)
18078: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
18079: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
18080: Procedure AddLoader( Loader : TBitmapLoader)
18081: Procedure ADDPARAM( VALUE : TPARAM)
18082: Procedure AddPassword( const Password : string)
18083: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
18084: Procedure AddState( oState : TniRegularExpressionState)
18085: Procedure AddStrings( Strings : TStrings);
18086: procedure AddStrings(Strings: TStrings);
18087: Procedure AddStrings1( Strings : TWideStrings);
18088: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
18089: Procedure AddToRecentDocs( const Filename : string)
18090: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
18091: Procedure AllFunctionsList1Click( Sender : TObject)
18092: procedure AllObjectsList1Click(Sender: TObject);
18093: Procedure Allocate( AAllocateBytes : Integer)
18094: procedure AllResourceList1Click(Sender: TObject);
18095: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
18096: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
18097: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
18098: Procedure AnsiFree( var s : AnsiString)
18099: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
18100: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
18101: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
18102: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
18103: Procedure AntiFreeze;
18104: Procedure APPEND
18105: Procedure Append( const S : WideString)
18106: procedure Append(S: string);
18107: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
18108: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
18109: Procedure AppendChunk( Val : OleVariant)
18110: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
18111: Procedure AppendStr( var Dest : string; S : string)
18112: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
18113: Procedure ApplyRange
18114: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18115: Procedure Arrange( Code : TListArrangement)
18116: procedure Assert(expr : Boolean; const msg: string);
18117: procedure Assert2(expr : Boolean; const msg: string);
18118: Procedure Assign( AList : TCustomBucketList)
18119: Procedure Assign( Other : TObject)
18120: Procedure Assign( Source : TDragObject)
18121: Procedure Assign( Source : TPersistent)
18122: Procedure Assign(Source: TPersistent)
18123: procedure Assign2(mystring, mypath: string);
18124: Procedure AssignCurValues( Source : TDataSet);
18125: Procedure AssignCurValues1( const CurValues : Variant);
18126: Procedure ASSIGNFIELD( FIELD : TFIELD)
18127: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
18128: Procedure AssignFile(var F: Text; FileName: string)
18129: procedure AssignFile(var F: TextFile; FileName: string)
18130: procedure AssignFileRead(var mystring, myfilename: string);
18131: procedure AssignFileWrite(mystring, myfilename: string);
18132: Procedure AssignTo( Other : TObject)
18133: Procedure AssignValues( Value : TParameters)
18134: Procedure ASSIGNVALUES( VALUE : TPARAMS)
18135: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
18136: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
18137: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
```

```
18138: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
18139: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18140: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
18141: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
18142: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
18143: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
18144: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
18145: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
18146: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
18147: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
18148: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18149: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
18150: procedure Beep
18151: Procedure BeepOk
18152: Procedure BeepQuestion
18153: Procedure BeepHand
18154: Procedure BeepExclamation
18155: Procedure BeepAsterisk
18156: Procedure BeepInformation
18157: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
18158: Procedure BeginLayout
18159: Procedure BeginTimer( const Delay, Resolution : Cardinal)
18160: Procedure BeginUpdate
18161: procedure BeginUpdate;
18162: procedure BigScreen1Click(Sender: TObject);
18163: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
18164: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
18165: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
18166: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
18167: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
18168: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
18169: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
18170: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
18171: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
18172: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
18173: Procedure BreakPointMenuClick( Sender : TObject)
18174: procedure BRINGTOFRONT
18175: procedure BringToFront;
18176: Procedure btnBackClick( Sender : TObject)
18177: Procedure btnBrowseClick( Sender : TObject)
18178: Procedure BtnClick( Index : TNavigateBtn)
18179: Procedure btnLargeIconsClick( Sender : TObject)
18180: Procedure BuildAndSendRequest( AURI : TIdURI)
18181: Procedure BuildCache
18182: Procedure BurnMemory( var Buff, BuffLen : integer)
18183: Procedure BurnMemoryStream( Destructo : TMemoryStream)
18184: Procedure CalculateFirstSet
18185: Procedure Cancel
18186: procedure CancelDrag;
18187: Procedure CancelEdit
18188: procedure CANCELHINT
18189: Procedure CancelRange
18190: Procedure CancelUpdates
18191: Procedure CancelWriteBuffer
18192: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
18193: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
18194: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
18195: procedure CaptureScreenFormat(vname: string; vextension: string);
18196: procedure CaptureScreenPNG(vname: string);
18197: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
18198: procedure CASCADE
18199: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
18200: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
18201: Procedure cbPathClick( Sender : TObject)
18202: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18203: Procedure cedebugAfterExecute( Sender : TPSScript)
18204: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18205: Procedure cedebugCompile( Sender : TPSScript)
18206: Procedure cedebugExecute( Sender : TPSScript)
18207: Procedure cedebugIdle( Sender : TObject)
18208: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18209: Procedure CenterHeight( const pc, pcParent : TControl)
18210: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18211: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18212: Procedure Change
18213: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18214: Procedure Changed
18215: Procedure ChangeDir( const ADirName : string)
18216: Procedure ChangeDirUp
18217: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
18218: Procedure ChangeLevelBy( Value : TChangeRange)
18219: Procedure ChDir(const s: string)
18220: Procedure Check(Status: Integer)
18221: Procedure CheckCommonControl( CC : Integer)
18222: Procedure CHECKFIELDNAME( const FIELDNAME : String)
18223: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
18224: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
18225: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
18226: Procedure CheckToken( T : Char)
```

```
18227: procedure CheckToken(t:char)
18228: Procedure CheckTokenSymbol( const S : string)
18229: procedure CheckTokenSymbol(s:string)
18230: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
18231: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18232: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
18233: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
18234: procedure CipherFile1Click(Sender: TObject);
18235: Procedure Clear;
18236: Procedure Clear1Click( Sender : TObject)
18237: Procedure ClearColor( Color : TColor)
18238: Procedure CLEARITEM( AITEM : TMENUITEM)
18239: Procedure ClearMapping
18240: Procedure ClearSelection( KeepPrimary : Boolean)
18241: Procedure ClearWriteBuffer
18242: Procedure Click
18243: Procedure Close
18244: Procedure Close1Click( Sender : TObject)
18245: Procedure CloseDatabase( Database : TDatabase)
18246: Procedure CloseDataSets
18247: Procedure CloseDialog
18248: Procedure CloseFile(var F: Text);
18249: Procedure Closure
18250: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18251: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18252: Procedure CodeCompletionList1Click( Sender : TObject)
18253: Procedure ColEnter
18254: Procedure Collapse
18255: Procedure Collapse( Recurse : Boolean)
18256: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
18257: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
18258: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
18259: Procedure Compile1Click( Sender : TObject)
18260: procedure ComponentCount1Click(Sender: TObject);
18261: Procedure Compress(azipfolder, azipfile: string)
18262: Procedure DeCompress(azipfolder, azipfile: string)
18263: Procedure XZip(azipfolder, azipfile: string)
18264: Procedure XUnZip(azipfolder, azipfile: string)
18265: Procedure Connect( const ATimeout : Integer)
18266: Procedure Connect( Socket : TSocket)
18267: procedure Console1Click(Sender: TObject);
18268: Procedure Continue
18269: Procedure ContinueCount( var Counter : TJclCounter)
18270: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18271: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
18272: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
18273: Procedure ConvertImage(vsource, vdestination: string);
18274: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
18275: Procedure ConvertBitmap(vsource, vdestination: string);
18276: Procedure ConvertToGray(Cnv: TCanvas);
18277: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
18278: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
18279: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
18280: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
18281: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
18282: Procedure CopyFrom( mbCopy : TMyBigInt)
18283: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
18284: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
18285: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
18286: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
18287: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
18288: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
18289: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
18290: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
18291: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
18292: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18293: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
18294: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18295: Procedure CopyToClipboard
18296: Procedure CountParts
18297: Procedure CreateDataSet
18298: Procedure CreateEmptyFile( const FileName : string)
18299: Procedure CreateFileFromString( const FileName, Data : string)
18300: Procedure CreateFromDelta( Source : TPacketDataSet)
18301: procedure CREATEHANDLE
18302: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18303: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
18304: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
18305: Procedure CreateTable
18306: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
18307: procedure CSyntax1Click(Sender: TObject);
18308: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
18309: Procedure CURSORPOSCHANGED
18310: procedure CutFirstDirectory(var S: String)
18311: Procedure DataBaseError(const Message: string)
18312: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
```

```
18313: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
18314: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
18315: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18316: Procedure DBIError(errorCode: Integer)
18317: Procedure DebugOutput( const AText : string)
18318: Procedure DebugRun1Click( Sender : TObject)
18319: procedure Dec;
18320: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
18321: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
18322: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
18323: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
18324: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
18325: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
18326: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18327: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
18328: Procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
18329: Procedure Decompile1Click( Sender : TObject)
18330: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18331: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18332: Procedure DeferLayout
18333: Procedure defFileread
18334: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
18335: Procedure DelayMicroseconds( const MicroSeconds : Integer)
18336: Procedure Delete
18337: Procedure Delete( const AFilename : string)
18338: Procedure Delete( const Index : Integer)
18339: Procedure DELETE( INDEX : INTEGER)
18340: Procedure Delete( Index : LongInt)
18341: Procedure Delete( Node : TTreeNode)
18342: procedure Delete(var s: AnyString; ifrom, icount: Longint);
18343: Procedure DeleteAlias( const Name : string)
18344: Procedure DeleteDriver( const Name : string)
18345: Procedure DeleteIndex( const Name : string)
18346: Procedure DeleteKey( const Section, Ident : String)
18347: Procedure DeleteRecords
18348: Procedure DeleteRecords( AffectRecords : TAffectRecords)
18349: Procedure DeleteString( var pStr : String; const pDelStr : string)
18350: Procedure DeleteTable
18351: procedure DelphiSite1Click(Sender: TObject);
18352: Procedure Deselect
18353: Procedure Deselect( Node : TTreeNode)
18354: procedure DestroyComponents
18355: Procedure DestroyHandle
18356: Procedure Diff( var X : array of Double)
18357: procedure Diff(var X: array of Double);
18358: Procedure DirCreate( const DirectoryName : String)');
18359: procedure DISABLEALIGN
18360: Procedure DisableConstraints
18361: Procedure Disconnect
18362: Procedure Disconnect( Socket : TSocket)
18363: Procedure Dispose
18364: procedure Dispose(P: PChar)
18365: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
18366: Procedure DoKey( Key : TDBCtrlGridKey)
18367: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18368: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18369: Procedure Dormant
18370: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
18371: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
18372: Procedure DoubleToComp( Value : Double; var Result : Comp)
18373: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18374: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
18375: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18376: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18377: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18378: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
18379: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18380: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
18381: procedure DrawFocusRect(const Rect: TRect);
18382: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
18383: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18384: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
18385: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
18386: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18387: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
18388: Procedure DropConnections
18389: Procedure DropDown
18390: Procedure DumpDescription( oStrings : TStrings)
18391: Procedure DumpStateTable( oStrings : TStrings)
18392: Procedure EDIT
18393: Procedure EditButtonClick
18394: Procedure EditFont1Click( Sender : TObject)
18395: procedure Ellipse(X1, Y1, X2, Y2: Integer);
18396: Procedure Ellipse1( const Rect : TRect);
18397: Procedure EMMS
18398: Procedure Encode( ADest : TStream)
18399: procedure ENDDRAG(DROP:BOOLEAN)
```

```
18400: Procedure EndEdit( Cancel : Boolean)
18401: Procedure EndTimer
18402: Procedure EndUpdate
18403: Procedure EraseSection( const Section : string)
18404: Procedure Error( const Ident : string)
18405: procedure Error(Ident:Integer)
18406: Procedure ErrorFmt( const Ident : string; const Args : array of const)
18407: Procedure ErrorStr( const Message : string)
18408: procedure ErrorStr(Message:String)
18409: Procedure Exchange( Index1, Index2 : Integer)
18410: procedure Exchange(Index1, Index2: Integer);
18411: Procedure Exec( FileName, Parameters, Directory : string)
18412: Procedure ExecProc
18413: Procedure ExecSQL( UpdateKind : TUpdateKind)
18414: Procedure Execute
18415: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
18416: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
18417: Procedure ExecuteCommand(executeFile, paramstring: string)
18418: Procedure ExecuteShell(executeFile, paramstring: string)
18419: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18420: Procedure ExitThread(ExitCode: Integer); stdcall;
18421: Procedure ExitProcess(ExitCode: Integer); stdcall;
18422: Procedure Expand( AUserName : String; AResults : TStrings)
18423: Procedure Expand( Recurse : Boolean)
18424: Procedure ExportClipboard1Click( Sender : TObject)
18425: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
18426: Procedure ExtractContentFields( Strings : TStrings)
18427: Procedure ExtractCookieFields( Strings : TStrings)
18428: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
18429: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18430: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18431: Procedure ExtractQueryFields( Strings : TStrings)
18432: Procedure FastDegToGrad
18433: Procedure FastDegToRad
18434: Procedure FastGradToDeg
18435: Procedure FastGradToRad
18436: Procedure FastRadToDeg
18437: Procedure FastRadToGrad
18438: Procedure FileClose( Handle : Integer)
18439: Procedure FileClose(handle: integer)
18440: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18441: Procedure FileStructure( AStructure : TIdFTPDataStructure)
18442: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
18443: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
18444: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
18445: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18446: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18447: Procedure FillIPList
18448: procedure FillRect(const Rect: TRect);
18449: Procedure FillTStrings( AStrings : TStrings)
18450: Procedure FilterOnBookmarks( Bookmarks : array of const)
18451: procedure FinalizePackage(Module: HMODULE)
18452: procedure FindClose;
18453: procedure FindClose2(var F: TSearchRec)
18454: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
18455: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
18456: Procedure FindNearest( const KeyValues : array of const)
18457: Procedure FinishContext
18458: Procedure FIRST
18459: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
18460: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18461: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
18462: Procedure FlushSchemaCache( const TableName : string)
18463: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
18464: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
18465: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
18466: Procedure FormActivate( Sender : TObject)
18467: procedure FormatLn(const format: String; const args: array of const);  //alias
18468: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18469: Procedure FormCreate( Sender : TObject)
18470: Procedure FormDestroy( Sender : TObject)
18471: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18472: procedure FormOutput1Click(Sender: TObject);
18473: Procedure FormToHtml( Form : TForm; Path : string)
18474: procedure FrameRect(const Rect: TRect);
18475: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18476: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
18477: Procedure Free( Buffer : TRecordBuffer)
18478: Procedure Free( Buffer : TValueBuffer)
18479: Procedure Free;
18480: Procedure FreeAndNil(var Obj:TObject)
18481: Procedure FreeImage
18482: procedure FreeMem(P: PChar; Size: Integer)
18483: Procedure FreeTreeData( Tree : TUpdateTree)
18484: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
18485: Procedure FullCollapse
18486: Procedure FullExpand
```

```
18487: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
18488: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
18489: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
18490: Procedure Get1( AURL : string; const AResponseContent : TStream);
18491: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
18492: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
18493: Procedure GetAliasNames( List : TStrings)
18494: Procedure GetAliasParams( const AliasName : string; List : TStrings)
18495: Procedure GetApplicationsRunning( Strings : TStrings)
18496: Procedure getBox(aURL, extension: string);
18497: Procedure GetCommandTypes( List : TWideStrings)
18498: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
18499: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
18500: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
18501: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
18502: Procedure GetDatabaseNames( List : TStrings)
18503: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
18504: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
18505: Procedure GetDir(d: byte; var s: string)
18506: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
18507: Procedure GetDriverNames( List : TStrings)
18508: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
18509: Procedure GetDriverParams( const DriverName : string; List : TStrings)
18510: Procedure GetEMails1Click( Sender : TObject)
18511: Procedure getEnvironmentInfo;
18512: Function getEnvironmentString: string;
18513: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
18514: Procedure GetFieldNames( const TableName : string; List : TStrings)
18515: Procedure GetFieldNames( const TableName : string; List : TStrings);
18516: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
18517: Procedure GETFIELDNAMES( LIST : TSTRINGS)
18518: Procedure GetFieldNames1( const TableName : string; List : TStrings);
18519: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
18520: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
18521: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
18522: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
18523: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
18524: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
18525: Procedure GetFormatSettings
18526: Procedure GetFromDIB( var DIB : TBitmapInfo)
18527: Procedure GetFromHDIB( HDIB : HBitmap)
18528: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
18529: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
18530: Procedure GetIcon( Index : Integer; Image : TIcon);
18531: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
18532: Procedure GetIndexInfo( IndexName : string)
18533: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
18534: Procedure GetIndexNames( List : TStrings)
18535: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
18536: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
18537: Procedure GetIndexNames4( const TableName : string; List : TStrings);
18538: Procedure GetInternalResponse
18539: Procedure GETITEMNAMES( LIST : TSTRINGS)
18540: procedure GetMem(P: PChar; Size: Integer)
18541: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
18542: procedure GetPackageDescription(ModuleName: PChar): string)
18543: Procedure GetPackageNames( List : TStrings);
18544: Procedure GetPackageNames1( List : TWideStrings);
18545: Procedure GetParamList( List : TList; const ParamNames : WideString)
18546: Procedure GetProcedureNames( List : TStrings);
18547: Procedure GetProcedureNames( List : TWideStrings);
18548: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
18549: Procedure GetProcedureNames1( List : TStrings);
18550: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
18551: Procedure GetProcedureNames3( List : TWideStrings);
18552: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
18553: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
18554: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
18555: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
18556: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
18557: Procedure GetProviderNames( Names : TWideStrings);
18558: Procedure GetProviderNames( Proc : TGetStrProc)
18559: Procedure GetProviderNames1( Names : TStrings);
18560: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
18561: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
18562: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
18563: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
18564: Procedure GetSchemaNames( List : TStrings);
18565: Procedure GetSchemaNames1( List : TWideStrings);
18566: Procedure getScriptandRunAsk;
18567: Procedure getScriptandRun(ascript: string);
18568: Procedure getScript(ascript: string);   //alias
18569: Procedure getWebScript(ascript: string);   //alias
18570: Procedure GetSessionNames( List : TStrings)
18571: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
18572: Procedure GetStrings( List : TStrings)
18573: Procedure GetSystemTime; stdcall;
18574: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
```

```
18575: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
18576: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
18577: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
18578: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
18579: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
18580: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
18581: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
18582: Procedure GetVisibleWindows( List : Tstrings)
18583: Procedure GoBegin
18584: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
18585: Procedure GotoCurrent( Table : TTable)
18586: procedure GotoEnd1Click(Sender: TObject);
18587: Procedure GotoNearest
18588: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
18589: Procedure HandleException( E : Exception; var Handled : Boolean)
18590: procedure HANDLEMESSAGE
18591: procedure HandleNeeded;
18592: Procedure Head( AURL : string)
18593: Procedure Help( var AHelpContents : TStringList; ACommand : String)
18594: Procedure HexToBinary( Stream : TStream)
18595: procedure HexToBinary(Stream:TStream)
18596: Procedure HideDragImage
18597: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
18598: Procedure HideTraybar
18599: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18600: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18601: Procedure HookOSExceptions
18602: Procedure HookSignal( RtlSigNum : Integer)
18603: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
18604: Procedure HTMLSyntax1Click( Sender : TObject)
18605: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18606: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
18607: Procedure ImportfromClipboard1Click( Sender : TObject)
18608: Procedure ImportfromClipboard2Click( Sender : TObject)
18609: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
18610: procedure Incb(var x: byte);
18611: Procedure Include1Click( Sender : TObject)
18612: Procedure IncludeOFF;  //preprocessing
18613: Procedure IncludeON;
18614: procedure Info1Click(Sender: TObject);
18615: Procedure InitAltRecBuffers( CheckModified : Boolean)
18616: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18617: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
18618: Procedure InitData( ASource : TDataSet)
18619: Procedure InitDelta( ADelta : TPacketDataSet);
18620: Procedure InitDelta1( const ADelta : OleVariant);
18621: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18622: Procedure Initialize
18623: procedure InitializePackage(Module: HMODULE)
18624: Procedure INITIATEACTION
18625: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
18626: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18627: Procedure InitModule( AModule : TComponent)
18628: Procedure InitStdConvs
18629: Procedure InitTreeData( Tree : TUpdateTree)
18630: Procedure INSERT
18631: Procedure Insert( Index : Integer; AClass : TClass)
18632: Procedure Insert( Index : Integer; AComponent : TComponent)
18633: Procedure Insert( Index : Integer; AObject : TObject)
18634: Procedure Insert( Index : Integer; const S : WideString)
18635: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18636: Procedure Insert(Index: Integer; const S: string);
18637: procedure Insert(Index: Integer; S: string);
18638: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18639: procedure InsertComponent(AComponent:TComponent)
18640: procedure InsertControl(AControl: TControl);
18641: Procedure InsertIcon( Index : Integer; Image : TIcon)
18642: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
18643: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18644: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18645: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18646: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18647: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18648: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
18649: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18650: Procedure InvalidateModuleCache
18651: Procedure InvalidateTitles
18652: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18653: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18654: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
18655: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18656: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18657: procedure JavaSyntax1Click(Sender: TObject);
18658: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
18659: Procedure KillDataChannel
18660: Procedure Largefont1Click( Sender : TObject)
18661: Procedure LAST
```

```
18662: Procedure LaunchCpl( FileName : string)
18663: Procedure Launch( const AFile : string)
18664: Procedure LaunchFile( const AFile : string)
18665: Procedure LetFileList(FileList: TStringlist; apath: string);
18666: Procedure lineToNumber( xmemo : String; met : boolean)
18667: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
18668: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
18669: Procedure ListViewData( Sender : TObject; Item : TListItem)
18670: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
18671: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
18672: Procedure ListViewDblClick( Sender : TObject)
18673: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18674: Procedure ListDLLExports(const FileName: string; List: TStrings);
18675: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
18676: procedure LoadBytecode1Click(Sender: TObject);
18677: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
18678: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
18679: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
18680: Procedure LoadFromFile( AFileName : string)
18681: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
18682: Procedure LoadFromFile( const FileName : string)
18683: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
18684: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
18685: Procedure LoadFromFile( const FileName : WideString)
18686: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18687: Procedure LoadFromFile(const AFileName: string)
18688: procedure LoadFromFile(FileName: string);
18689: procedure LoadFromFile(FileName:String)
18690: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
18691: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
18692: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
18693: Procedure LoadFromStream( const Stream : TStream)
18694: Procedure LoadFromStream( S : TStream)
18695: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18696: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18697: Procedure LoadFromStream( Stream : TStream)
18698: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
18699: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
18700: procedure LoadFromStream(Stream: TStream);
18701: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
18702: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
18703: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
18704: Procedure LoadLastFile1Click( Sender : TObject)
18705: { LoadIcoToImage loads two icons from resource named NameRes,
18706:   into two image lists ALarge and ASmall}
18707: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
18708: Procedure LoadMemo
18709: Procedure LoadParamsFromIniFile( FFileName : WideString)
18710: Procedure Lock
18711: Procedure Login
18712: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
18713: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
18714: Procedure MakeCaseInsensitive
18715: Procedure MakeDeterministic( var bChanged : boolean)
18716: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
18717: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
18718: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
18719: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
18720: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18721: Procedure SetRectComplexFormatStr( const S : string)
18722: Procedure SetPolarComplexFormatStr( const S : string)
18723: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18724: Procedure MakeVisible
18725: Procedure MakeVisible( PartialOK : Boolean)
18726: Procedure Manual1Click( Sender : TObject)
18727: Procedure MarkReachable
18728: Procedure maXbox;  //shows the exe version data in a win box
18729: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
18730: Procedure Memo1Change( Sender : TObject)
18731: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
18732: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
18733: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18734: procedure Memory1Click(Sender: TObject);
18735: Procedure MERGE( MENU : TMAINMENU)
18736: Procedure MergeChangeLog
18737: procedure MINIMIZE
18738: Procedure MinimizeMaxbox;
18739: procedure MyCopyFile(Name1,Name2:string);
18740: Procedure MkDir(const s: string)
18741: Procedure MakeDir(const s: string)');
18742: Procedure ChangeDir(const s: string)');
18743: Function makeFile(const FileName: string): integer)');
18744: Procedure mnuPrintFont1Click( Sender : TObject)
18745: procedure ModalStarted
18746: Procedure Modified
```

```
18747: Procedure ModifyAlias( Name : string; List : TStrings)
18748: Procedure ModifyDriver( Name : string; List : TStrings)
18749: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
18750: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
18751: Procedure Move( CurIndex, NewIndex : Integer)
18752: procedure Move(CurIndex, NewIndex: Integer);
18753: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
18754: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
18755: Procedure moveCube( o : TMyLabel)
18756: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
18757: procedure MoveTo(X, Y: Integer);
18758: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
18759: Procedure MovePoint(var x,y:Extended; const angle:Extended);
18760: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
18761: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
18762: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
18763: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
18764: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
18765: procedure New(P: PChar)
18766: procedure New1Click(Sender: TObject);
18767: procedure NewInstance1Click(Sender: TObject);
18768: Procedure NEXT
18769: Procedure NextMonth
18770: Procedure Noop
18771: Procedure NormalizePath( var APath : string)
18772: procedure ObjectBinaryToText(Input, Output: TStream)
18773: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18774: procedure ObjectResourceToText(Input, Output: TStream)
18775: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18776: procedure ObjectTextToBinary(Input, Output: TStream)
18777: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18778: procedure ObjectTextToResource(Input, Output: TStream)
18779: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18780: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
18781: Procedure Open( const UserID : WideString; const Password : WideString);
18782: Procedure Open;
18783: Procedure open1Click( Sender : TObject)
18784: Procedure OpenCdDrive
18785: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
18786: Procedure OpenCurrent
18787: Procedure OpenFile(vfilenamepath: string)
18788: Procedure OpenDirectory1Click( Sender : TObject)
18789: Procedure OpenDir(adir: string);
18790: Procedure OpenIndexFile( const IndexName : string)
18791: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
18792: Procedure OpenWriteBuffer( const AThreshhold : Integer)
18793: Procedure OptimizeMem
18794: Procedure Options1( AURL : string);
18795: Procedure OutputDebugString(lpOutputString : PChar)
18796: Procedure PackBuffer
18797: Procedure Paint
18798: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
18799: Procedure PaintToTBitmap( Target : TBitmap)
18800: Procedure PaletteChanged
18801: Procedure ParentBiDiModeChanged
18802: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
18803: Procedure PasteFromClipboard;
18804: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
18805: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
18806: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
18807: Procedure PError( Text : string)
18808: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18809: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
18810: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
18811: procedure playmp3(mpath: string);
18812: Procedure PlayMP31Click( Sender : TObject)
18813: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
18814: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
18815: procedure PolyBezier(const Points: array of TPoint);
18816: procedure PolyBezierTo(const Points: array of TPoint);
18817: procedure Polygon(const Points: array of TPoint);
18818: procedure Polyline(const Points: array of TPoint);
18819: Procedure Pop
18820: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
18821: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
18822: Procedure POPUP( X, Y : INTEGER)
18823: Procedure PopupURL(URL : WideString);
18824: Procedure POST
18825: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
18826: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
18827: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
18828: Procedure PostUser( const Email, FirstName, LastName : WideString)
18829: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18830: procedure Pred(X: int64);
18831: Procedure Prepare
18832: Procedure PrepareStatement
18833: Procedure PreProcessXML( AList : TStrings)
18834: Procedure PreventDestruction
```

```
18835: Procedure Print( const Caption : string)
18836: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18837: procedure printf(const format: String; const args: array of const);
18838: Procedure PrintList(Value: TStringList);
18839: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
18840: Procedure Printout1Click( Sender : TObject)
18841: Procedure ProcessHeaders
18842: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
18843: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
18844: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
18845: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
18846: Procedure ProcessMessagesOFF;  //application.processmessages
18847: Procedure ProcessMessagesON;
18848: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
18849: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
18850: Procedure Proclist Size is: 3797 /1415
18851: Procedure procMessClick( Sender : TObject)
18852: Procedure PSScriptCompile( Sender : TPSScript)
18853: Procedure PSScriptExecute( Sender : TPSScript)
18854: Procedure PSScriptLine( Sender : TObject)
18855: Procedure Push( ABoundary : string)
18856: procedure PushItem(AItem: Pointer)
18857: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
18858: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
18859: procedure PutLinuxLines(const Value: string)
18860: Procedure Quit
18861: Procedure RaiseConversionError( const AText : string);
18862: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
18863: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
18864: procedure RaiseException(Ex: TIFException; Param: String);
18865: Procedure RaiseExceptionForLastCmdResult;
18866: procedure RaiseLastException;
18867: procedure RaiseException2;
18868: Procedure RaiseException3(const Msg: string);
18869: Procedure RaiseExcept(const Msg: string);
18870: Procedure RaiseLastOSError
18871: Procedure RaiseLastWin32;
18872: procedure RaiseLastWin32Error)
18873: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
18874: Procedure RandomFillStream( Stream : TMemoryStream)
18875: procedure randomize;
18876: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
18877: Procedure RCS
18878: Procedure Read( Socket : TSocket)
18879: Procedure ReadBlobData
18880: procedure ReadBuffer(Buffer:String;Count:LongInt)
18881: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
18882: Procedure ReadSection( const Section : string; Strings : TStrings)
18883: Procedure ReadSections( Strings : TStrings)
18884: Procedure ReadSections( Strings : TStrings);
18885: Procedure ReadSections1( const Section : string; Strings : TStrings);
18886: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
18887: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
18888: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
18889: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
18890: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
18891: Procedure Realign;
18892: procedure Rectangle(X1, Y1, X2, Y2: Integer);
18893: Procedure Rectangle1( const Rect : TRect);
18894: Procedure RectCopy( var Dest : TRect; const Source : TRect)
18895: Procedure RectFitToScreen( var R : TRect)
18896: Procedure RectGrow( var R : TRect; const Delta : Integer)
18897: Procedure RectGrowX( var R : TRect; const Delta : Integer)
18898: Procedure RectGrowY( var R : TRect; const Delta : Integer)
18899: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
18900: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
18901: Procedure RectNormalize( var R : TRect)
18902: //  TFileCallbackProcedure = procedure(filename:string);
18903: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
18904: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
18905: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
18906: Procedure Refresh;
18907: Procedure RefreshData( Options : TFetchOptions)
18908: Procedure REFRESHLOOKUPLIST
18909: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
18910: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
18911: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
18912: Procedure RegisterChanges( Value : TChangeLink)
18913: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
18914: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
18915: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
18916: Procedure ReInitialize( ADelay : Cardinal)
18917: procedure RELEASE
18918: Procedure Remove( const AByteCount : integer)
18919: Procedure REMOVE( FIELD : TFIELD)
18920: Procedure REMOVE( ITEM : TMENUITEM)
18921: Procedure REMOVE( POPUP : TPOPUPMENU)
18922: Procedure RemoveAllPasswords
18923: procedure RemoveComponent(AComponent:TComponent)
```

```
18924: Procedure RemoveDir( const ADirName : string)
18925: Procedure RemoveLambdaTransitions( var bChanged : boolean)
18926: Procedure REMOVEPARAM( VALUE : TPARAM)
18927: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
18928: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
18929: Procedure Rename( const ASourceFile, ADestFile : string)
18930: Procedure Rename( const FileName : string; Reload : Boolean)
18931: Procedure RenameTable( const NewTableName : string)
18932: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
18933: Procedure Replace1Click( Sender : TObject)
18934: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
18935: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
18936: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
18937: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
18938: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
18939: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
18940: Procedure Requery( Options : TExecuteOptions)
18941: Procedure Reset
18942: Procedure Reset1Click( Sender : TObject)
18943: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
18944: procedure ResourceExplore1Click(Sender: TObject);
18945: Procedure RestoreContents
18946: Procedure RestoreDefaults
18947: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
18948: Procedure RetrieveHeaders
18949: Procedure RevertRecord
18950: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
18951: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18952: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18953: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
18954: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
18955: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
18956: Procedure RleCompress2( Stream : TStream)
18957: Procedure RleDecompress2( Stream : TStream)
18958: Procedure RmDir(const S: string)
18959: Procedure Rollback
18960: Procedure Rollback( TransDesc : TTransactionDesc)
18961: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
18962: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
18963: Procedure RollbackTrans
18964: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
18965: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
18966: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
18967: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
18968: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
18969: Procedure S_EBox( const AText : string)
18970: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
18971: Procedure S_IBox( const AText : string)
18972: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
18973: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
18974: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
18975: Procedure SampleVarianceAndMean
18976: ( const X : TDynFloatArray; var Variance, Mean : Float)
18977: Procedure Save2Click( Sender : TObject)
18978: Procedure Saveas3Click( Sender : TObject)
18979: Procedure Savebefore1Click( Sender : TObject)
18980: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
18981: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18982: Procedure SaveConfigFile
18983: Procedure SaveOutput1Click( Sender : TObject)
18984: procedure SaveScreenshotClick(Sender: TObject);
18985: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
18986: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
18987: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
18988: Procedure SaveToFile( AFileName : string)
18989: Procedure SAVETOFILE( const FILENAME : String)
18990: Procedure SaveToFile( const FileName : WideString)
18991: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
18992: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18993: procedure SaveToFile(FileName: string);
18994: procedure SaveToFile(FileName:String)
18995: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
18996: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18997: Procedure SaveToStream( S : TStream)
18998: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18999: Procedure SaveToStream( Stream : TStream)
19000: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
19001: procedure SaveToStream(Stream: TStream);
19002: procedure SaveToStream(Stream:TStream)
19003: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
19004: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
19005: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
19006: procedure Say(const sText: string)
19007: Procedure SBytecode1Click( Sender : TObject)
19008: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
19009: procedure ScriptExplorer1Click(Sender: TObject);
19010: Procedure Scroll( Distance : Integer)
19011: Procedure Scroll( DX, DY : Integer)
```

```
19012: procedure ScrollBy(DeltaX, DeltaY: Integer);
19013: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
19014: Procedure ScrollTabs( Delta : Integer)
19015: Procedure Search1Click( Sender : TObject)
19016: procedure SearchAndOpenDoc(vfilenamepath: string)
19017: procedure SearchAndOpenFile(vfilenamepath: string)
19018: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
19019: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19020: Procedure SearchNext1Click( Sender : TObject)
19021: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
19022: Procedure Select1( const Nodes : array of TTreeNode);
19023: Procedure Select2( Nodes : TList);
19024: Procedure SelectNext( Direction : Boolean)
19025: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
19026: Procedure SelfTestPEM  //unit uPSI_cPEM
19027: Procedure Send( AMsg : TIdMessage)
19028: //config forst in const MAILINIFILE = 'maildef.ini';
19029: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
19030: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19031: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19032: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
19033: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
19034: Procedure SendResponse
19035: Procedure SendStream( AStream : TStream)
19036: Procedure Set8087CW( NewCW : Word)
19037: Procedure SetAll( One, Two, Three, Four : Byte)
19038: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
19039: Procedure SetAppDispatcher( const ADispatcher : TComponent)
19040: procedure SetArrayLength;
19041: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
19042: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19043: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer);  //all init
19044: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
19045: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);');
19046: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);');
19047: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
19048: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
19049: Procedure SetAsHandle( Format : Word; Value : THandle)
19050: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
19051: procedure SetCaptureControl(Control: TControl);
19052: Procedure SetColumnAttributes
19053: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
19054: Procedure SetCustomHeader( const Name, Value : string)
19055: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
19056: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
19057: Procedure SetFocus
19058: procedure SetFocus; virtual;
19059: Procedure SetInitialState
19060: Procedure SetKey
19061: procedure SetLastError(ErrorCode: Integer)
19062: procedure SetLength;
19063: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
19064: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
19065: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
19066: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
19067: Procedure SetParams1( UpdateKind : TUpdateKind);
19068: Procedure SetPassword( const Password : string)
19069: Procedure SetPointer( Ptr : Pointer; Size : Longint)
19070: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
19071: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
19072: Procedure SetProvider( Provider : TComponent)
19073: Procedure SetProxy( const Proxy : string)
19074: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
19075: Procedure SetRange( const StartValues, EndValues : array of const)
19076: Procedure SetRangeEnd
19077: Procedure SetRate( const aPercent, aYear : integer)
19078: procedure SetRate(const aPercent, aYear: integer)
19079: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19080: Procedure SetSafeCallExceptionMsg( Msg : String)
19081: procedure SETSELTEXTBUF(BUFFER:PCHAR)
19082: Procedure SetSize( AWidth, AHeight : Integer)
19083: procedure SetSize(NewSize:LongInt)
19084: procedure SetString(var s: string; buffer: PChar; len: Integer)
19085: Procedure SetStrings( List : TStrings)
19086: Procedure SetText( Text : PwideChar)
19087: procedure SetText(Text: PChar);
19088: Procedure SetTextBuf( Buffer : PChar)
19089: procedure SETTEXTBUF(BUFFER:PCHAR)
19090: Procedure SetTick( Value : Integer)
19091: Procedure SetTimeout( ATimeOut : Integer)
19092: Procedure SetTraceEvent( Event : TDBXTraceEvent)
19093: Procedure SetUserName( const UserName : string)
19094: Procedure SetWallpaper( Path : string);
19095: procedure ShellStyle1Click(Sender: TObject);
19096: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
19097: Procedure ShowFileProperties( const FileName : string)
19098: Procedure ShowInclude1Click( Sender : TObject)
19099: Procedure ShowInterfaces1Click( Sender : TObject)
```

```
19100: Procedure ShowLastException1Click( Sender : TObject)
19101: Procedure ShowMessage( const Msg : string)
19102: Procedure ShowMessageBig(const aText : string);
19103: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
19104: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
19105: Procedure MsgBig(const aText : string);        //alias
19106: procedure showmessage(mytext: string);
19107: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
19108: procedure ShowMessageFmt(const Msg: string; Params: array of const))
19109: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
19110: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
19111: Procedure ShowSearchDialog( const Directory : string)
19112: Procedure ShowSpecChars1Click( Sender : TObject)
19113: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
19114: Procedure ShredFile( const FileName : string; Times : Integer)
19115: procedure Shuffle(vQ: TStringList);
19116: Procedure ShuffleList( var List : array of Integer; Count : Integer)
19117: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
19118: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
19119: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
19120: Procedure Site( const ACommand : string)
19121: Procedure SkipEOL
19122: Procedure Sleep( ATime : cardinal)
19123: Procedure Sleep( milliseconds : Cardinal)
19124: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
19125: Procedure Slinenumbers1Click( Sender : TObject)
19126: Procedure Sort
19127: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
19128: procedure Speak(const sText: string)   //async like voice
19129: procedure Speak2(const sText: string)  //sync
19130: procedure Split(Str: string;  SubStr: string; List: TStrings);
19131: Procedure SplitNameValue( const Line : string; var Name, Value : string)
19132: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
19133: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
19134: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
19135: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
19136: procedure SQLSyntax1Click(Sender: TObject);
19137: Procedure SRand( Seed : RNG_IntType)
19138: Procedure Start
19139: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
19140: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
19141: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
19142: Procedure StartTransaction( TransDesc : TTransactionDesc)
19143: Procedure Status( var AStatusList : TStringList)
19144: Procedure StatusBar1DblClick( Sender : TObject)
19145: Procedure StepInto1Click( Sender : TObject)
19146: Procedure StepIt
19147: Procedure StepOut1Click( Sender : TObject)
19148: Procedure Stop
19149: procedure stopmp3;
19150: procedure StartWeb(aurl: string);
19151: Procedure Str(aint: integer; astr: string); //of system
19152: Procedure StrDispose( Str : PChar)
19153: procedure StrDispose(Str: PChar)
19154: Procedure StrReplace(var Str: String; Old, New: String);
19155: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
19156: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
19157: Procedure StringToBytes( Value : String; Bytes : array of byte)
19158: procedure StrSet(c : Char; I : Integer; var s : String);
19159: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19160: Procedure StructureMount( APath : String)
19161: procedure STYLECHANGED(SENDER:TOBJECT)
19162: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
19163: procedure Succ(X: int64);
19164: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
19165: procedure SwapChar(var X,Y: char);  //swapX follows
19166: Procedure SwapFloats( var X, Y : Float)
19167: procedure SwapGrid(grd: TStringGrid);
19168: Procedure SwapOrd( var I, J : Byte);
19169: Procedure SwapOrd( var X, Y : Integer)
19170: Procedure SwapOrd1( var I, J : Shortint);
19171: Procedure SwapOrd2( var I, J : Smallint);
19172: Procedure SwapOrd3( var I, J : Word);
19173: Procedure SwapOrd4( var I, J : Integer);
19174: Procedure SwapOrd5( var I, J : Cardinal);
19175: Procedure SwapOrd6( var I, J : Int64);
19176: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
19177: Procedure Synchronize1( Method : TMethod);
19178: procedure SyntaxCheck1Click(Sender: TObject);
19179: Procedure SysFreeString(const S: WideString); stdcall;
19180: Procedure TakeOver( Other : TLinearBitmap)
19181: Procedure Talkln(const sText: string)  //async voice
19182: procedure tbtn6resClick(Sender: TObject);
19183: Procedure tbtnUseCaseClick( Sender : TObject)
19184: procedure TerminalStyle1Click(Sender: TObject);
19185: Procedure Terminate
19186: Procedure texSyntax1Click( Sender : TObject)
19187: procedure TextOut(X, Y: Integer; Text: string);
19188: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
```

```
19189: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
19190: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
19191: Procedure TextStart
19192: procedure TILE
19193: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
19194: Procedure TitleClick( Column : TColumn)
19195: Procedure ToDo
19196: procedure toolbtnTutorialClick(Sender: TObject);
19197: Procedure Trace1( AURL : string; const AResponseContent : TStream);
19198: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
19199: Procedure Truncate
19200: procedure Tutorial101Click(Sender: TObject);
19201: procedure Tutorial10Statistics1Click(Sender: TObject);
19202: procedure Tutorial11Forms1Click(Sender: TObject);
19203: procedure Tutorial12SQL1Click(Sender: TObject);
19204: Procedure tutorial1Click( Sender : TObject)
19205: Procedure tutorial21Click( Sender : TObject)
19206: Procedure tutorial31Click( Sender : TObject)
19207: Procedure tutorial4Click( Sender : TObject)
19208: Procedure Tutorial5Click( Sender : TObject)
19209: procedure Tutorial6Click(Sender: TObject);
19210: procedure Tutorial91Click(Sender: TObject);
19211: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19212: procedure UniqueString(var str: AnsiString)
19213: procedure UnloadLoadPackage(Module: HMODULE)
19214: Procedure Unlock
19215: Procedure UNMERGE( MENU : TMAINMENU)
19216: Procedure UnRegisterChanges( Value : TChangeLink)
19217: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19218: Procedure UnregisterConversionType( const AType : TConvType)
19219: Procedure UnRegisterProvider( Prov : TCustomProvider)
19220: Procedure UPDATE
19221: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19222: Procedure UPDATECURSORPOS
19223: Procedure UpdateFile
19224: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19225: Procedure UpdateResponse( AResponse : TWebResponse)
19226: Procedure UpdateScrollBar
19227: Procedure UpdateView1Click( Sender : TObject)
19228: procedure Val(const s: string; var n, z: Integer)
19229: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19230: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
19231: Procedure VariantAdd( const src : Variant; var dst : Variant)
19232: Procedure VariantAnd( const src : Variant; var dst : Variant)
19233: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19234: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19235: Procedure VariantClear( var V : Variant)
19236: Procedure VariantCpy( const src : Variant; var dst : Variant)
19237: Procedure VariantDiv( const src : Variant; var dst : Variant)
19238: Procedure VariantMod( const src : Variant; var dst : Variant)
19239: Procedure VariantMul( const src : Variant; var dst : Variant)
19240: Procedure VariantOr( const src : Variant; var dst : Variant)
19241: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
19242: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
19243: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
19244: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
19245: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
19246: Procedure VariantShl( const src : Variant; var dst : Variant)
19247: Procedure VariantShr( const src : Variant; var dst : Variant)
19248: Procedure VariantSub( const src : Variant; var dst : Variant)
19249: Procedure VariantXor( const src : Variant; var dst : Variant)
19250: Procedure VarCastError;
19251: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19252: Procedure VarInvalidOp
19253: Procedure VarInvalidNullOp
19254: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19255: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19256: Procedure VarArrayCreateError
19257: Procedure VarResultCheck( AResult : HRESULT);
19258: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19259: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19260: Function VarTypeAsText( const AType : TVarType) : string
19261: procedure Voice(const sText: string) //async
19262: procedure Voice2(const sText: string) //sync
19263: Procedure WaitMiliSeconds( AMSec : word)
19264: Procedure WaitMS( AMSec : word)');
19265: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
19266: Procedure WideAppend( var dst : WideString; const src : WideString)
19267: Procedure WideAssign( var dst : WideString; var src : WideString)
19268: Procedure WideDelete( var dst : WideString; index, count : Integer)
19269: Procedure WideFree( var s : WideString)
19270: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19271: Procedure WideFromPChar( var dst : WideString; src : PChar)
19272: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19273: Procedure WideSetLength( var dst : WideString; len : Integer)
19274: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
19275: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
19276: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19277: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
```

```
19278: Procedure HttpGet(const Url: string; Stream:TStream);
19279: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
19280: Procedure WordWrap1Click( Sender : TObject)
19281: Procedure Write( const AOut : string)
19282: Procedure Write( Socket : TSocket)
19283: procedure Write(S: string);
19284: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19285: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19286: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19287: procedure WriteBuffer(Buffer:String;Count:LongInt)
19288: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19289: Procedure WriteChar( AValue : Char)
19290: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19291: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19292: Procedure WriteFloat( const Section, Name : string; Value : Double)
19293: Procedure WriteHeader( AHeader : TStrings)
19294: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19295: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19296: Procedure WriteLn( const AOut : string)
19297: procedure Writeln(s: string);
19298: Procedure WriteLog( const FileName, LogLine : string)
19299: Procedure WriteRFCReply( AReply : TIdRFCReply)
19300: Procedure WriteRFCStrings( AStrings : TStrings)
19301: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19302: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
19303: Procedure WriteString( const Section, Ident, Value : String)
19304: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19305: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19306: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19307: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19308: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19309: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
19310: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19311: procedure XMLSyntax1Click(Sender: TObject);
19312: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19313: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19314: Procedure ZeroFillStream( Stream : TMemoryStream)
19315: procedure XMLSyntax1Click(Sender: TObject);
19316: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19317: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19318: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19319: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19320: procedure(Sender, Source: TObject;X, Y: Integer)
19321: procedure(Sender, Target: TObject; X, Y: Integer)
19322: procedure(Sender: TObject; ASection, AWidth: Integer)
19323: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
19324: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
19325: procedure(Sender: TObject; var Action: TCloseAction)
19326: procedure(Sender: TObject; var CanClose: Boolean)
19327: procedure(Sender: TObject; var Key: Char);
19328: ProcedureName ProcedureNames ProcedureParametersCursor @
19329:
19330: *************Now Constructors constructor *************
19331:  Size is: 1248 1115 996 628 550 544 501 459 (381)
19332:  Attach( VersionInfoData : Pointer; Size : Integer)
19333:  constructor Create( ABuckets : TBucketListSizes)
19334:  Create( ACallBackWnd : HWND)
19335:  Create( AClient : TCustomTaskDialog)
19336:  Create( AClient : TIdTelnet)
19337:  Create( ACollection : TCollection)
19338:  Create( ACollection : TFavoriteLinkItems)
19339:  Create( ACollection : TTaskDialogButtons)
19340:  Create( AConnection : TIdCustomHTTP)
19341:  Create( ACreateSuspended : Boolean)
19342:  Create( ADataSet : TCustomSQLDataSet)
19343:  CREATE( ADATASET : TDATASET)
19344:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19345:  Create( AGrid : TCustomDBGrid)
19346:  Create( AGrid : TStringGrid; AIndex : Longint)
19347:  Create( AHTTP : TIdCustomHTTP)
19348:  Create( AListItems : TListItems)
19349:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19350:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19351:  Create( AOwner : TCommonCalendar)
19352:  Create( AOwner : TComponent)
19353:  CREATE( AOWNER : TCOMPONENT)
19354:  Create( AOwner : TCustomListView)
19355:  Create( AOwner : TCustomOutline)
19356:  Create( AOwner : TCustomRichEdit)
19357:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19358:  Create( AOwner : TCustomTreeView)
19359:  Create( AOwner : TIdUserManager)
19360:  Create( AOwner : TListItems)
19361:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19362:  CREATE( AOWNER : TPERSISTENT)
19363:  Create( AOwner : TPersistent)
19364:  Create( AOwner : TTable)
19365:  Create( AOwner : TTreeNodes)
19366:  Create( AOwner : TWinControl; const ClassName : string)
```

```
19367:  Create( AParent : TIdCustomHTTP)
19368:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
19369:  Create( AProvider : TBaseProvider)
19370:  Create( AProvider : TCustomProvider);
19371:  Create( AProvider : TDataSetProvider)
19372:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19373:  Create( ASocket : TSocket)
19374:  Create( AStrings : TWideStrings)
19375:  Create( AToolBar : TToolBar)
19376:  Create( ATreeNodes : TTreeNodes)
19377:  Create( Autofill : boolean)
19378:  Create( AWebPageInfo : TAbstractWebPageInfo)
19379:  Create( AWebRequest : TWebRequest)
19380:  Create( Collection : TCollection)
19381:  Create( Collection : TIdMessageParts; ABody : TStrings)
19382:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
19383:  Create( Column : TColumn)
19384:  Create( const AConvFamily : TConvFamily; const ADescription : string)
19385:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
19386:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
19387:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
19388:  Create( const ATabSet : TTabSet)
19389:  Create( const Compensate : Boolean)
19390:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
19391:  Create( const FileName : string)
19392:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
19393:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
19394:  Create( const MaskValue : string)
19395:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19396:  Create( const Prefix : string)
19397:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
19398:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19399:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19400:  Create( CoolBar : TCoolBar)
19401:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
19402:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
19403:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
19404:  Create( DBCtrlGrid : TDBCtrlGrid)
19405:  Create( DSTableProducer : TDSTableProducer)
19406:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
19407:  Create( ErrorCode : DBIResult)
19408:  Create( Field : TBlobField; Mode : TBlobStreamMode)
19409:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
19410:  Create( HeaderControl : TCustomHeaderControl)
19411:  Create( HTTPRequest : TWebRequest)
19412:  Create( iStart : integer; sText : string)
19413:  Create( iValue : Integer)
19414:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19415:  Create( MciErrNo : MCIERROR; const Msg : string)
19416:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19417:  Create( Message : string; ErrorCode : DBResult)
19418:  Create( Msg : string)
19419:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19420:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19421:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19422:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19423:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
19424:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19425:  Create( Owner : TCustomComboBoxEx)
19426:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19427:  Create( Owner : TPersistent)
19428:  Create( Params : TStrings)
19429:  Create( Size : Cardinal)
19430:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19431:  Create( StatusBar : TCustomStatusBar)
19432:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
19433:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19434:  Create(AHandle:Integer)
19435:  Create(AOwner: TComponent); virtual;
19436:  Create(const AURI : string)
19437:  Create(FileName:String;Mode:Word)
19438:  Create(Instance:THandle;ResName:String;ResType:PChar)
19439:  Create(Stream : TStream)
19440:  Create1( ADataset : TDataset);
19441:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
19442:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19443:  Create2( Other : TObject);
19444:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19445:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19446:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
19447:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19448:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19449:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19450:  CreateRes( Ident : Integer);
```

```
19451:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
19452:  CreateRes( ResStringRec : PResStringRec);
19453:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
19454:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19455:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19456:  CreateSize( AWidth, AHeight : Integer)
19457:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19458:
19459: ------------------------------------------------------------------------------
19460: unit uPSI_MathMax;
19461: ------------------------------------------------------------------------------
19462:  CONSTS
19463:   Bernstein: Float = 0.280169499023869133036436491230 7;  // Bernstein constant
19464:   Cbrt2: Float     = 1.259921049894873164767210607278 2;  // CubeRoot(2)
19465:   Cbrt3: Float     = 1.442249570307408382321638310780 1;  // CubeRoot(3)
19466:   Cbrt10: Float    = 2.154434690031883721759293566519 4;  // CubeRoot(10)
19467:   Cbrt100: Float   = 4.641588833612778892410076350919 4;  // CubeRoot(100)
19468:   CbrtPi: Float    = 1.464591887561523263020142527263 8;  // CubeRoot(PI)
19469:   Catalan: Float   = 0.915965594177219015054603514932 4;  // Catalan constant
19470:   PiJ:   Float       = 3.141592653589793238462643383279 5;  // PI
19471:   PI:  Extended =  3.141592653589793238462643383279 5);
19472:   PiOn2: Float     = 1.570796326794896619231321691639 8;  // PI / 2
19473:   PiOn3: Float     = 1.047197551196597746154214461093 2;  // PI / 3
19474:   PiOn4: Float     = 0.785398163397448309615660845819 88; // PI / 4
19475:   Sqrt2: Float     = 1.414213562373095048801688724209 7;  // Sqrt(2)
19476:   Sqrt3: Float     = 1.732050807568877293527446341505 9;  // Sqrt(3)
19477:   Sqrt5: Float     = 2.236067977499789696409173668731 3;  // Sqrt(5)
19478:   Sqrt10: Float    = 3.162277660168379331998893544432 7;  // Sqrt(10)
19479:   SqrtPi: Float    = 1.772453850905516027298167483341 1;  // Sqrt(PI)
19480:   Sqrt2Pi: Float   = 2.506628274631000502415765284811;   // Sqrt(2 * PI)
19481:   TwoPi: Float     = 6.283185307179586476925286766559;   // 2 * PI
19482:   ThreePi: Float   = 9.424777960769379715387930149838 5;  // 3 * PI
19483:   Ln2: Float       = 0.693147180559945309417232121458 18; // Ln(2)
19484:   Ln10: Float      = 2.302585092994045684017991454684 4;  // Ln(10)
19485:   LnPi: Float      = 1.144729885849400174143427351353 1;  // Ln(PI)
19486:   Log2J: Float     = 0.301029995663981195213738894724 49; // Log10(2)
19487:   Log3: Float      = 0.477121254719662437295027903255 12; // Log10(3)
19488:   LogPi: Float     = 0.497149872694133854351268288290 9;  // Log10(PI)
19489:   LogE: Float      = 0.434294481903251827651128918916 61; // Log10(E)
19490:   E: Float         = 2.718281828459045235360287471352 7;  // Natural constant
19491:   hLn2Pi: Float    = 0.918938533204672741780329736405 62; // Ln(2*PI)/2
19492:   inv2Pi: Float    = 0.159154943091895335768883763372514362034459645740 46; // 0.5/Pi
19493:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
19494:   GoldenMean: Float   = 1.618033988749894848204586834365638;  // GoldenMean
19495:   EulerMascheroni: Float = 0.577215664901532860606512090082 4;  // Euler GAMMA
19496:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
19497:   StDelta         : Extended = 0.00001;  {delta for difference equations}
19498:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
19499:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
19500:
19501: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
19502: begin
19503:   MetersPerInch = 0.0254; // [1]
19504:   MetersPerFoot = MetersPerInch * 12;
19505:   MetersPerYard = MetersPerFoot * 3;
19506:   MetersPerMile = MetersPerFoot * 5280;
19507:   MetersPerNauticalMiles = 1852;
19508:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
19509:   MetersPerLightSecond = 2.99792458E8; // [5]
19510:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
19511:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
19512:   MetersPerCubit = 0.4572; // [6][7]
19513:   MetersPerFathom = MetersPerFoot * 6;
19514:   MetersPerFurlong = MetersPerYard * 220;
19515:   MetersPerHand = MetersPerInch * 4;
19516:   MetersPerPace = MetersPerInch * 30;
19517:   MetersPerRod = MetersPerFoot * 16.5;
19518:   MetersPerChain = MetersPerRod * 4;
19519:   MetersPerLink = MetersPerChain / 100;
19520:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
19521:   MetersPerPica = MetersPerPoint * 12;
19522:
19523:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
19524:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
19525:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
19526:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
19527:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
19528:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
19529:
19530:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
19531:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
19532:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
19533:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
19534:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
19535:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
19536:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
19537:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
19538:
19539:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
```

```
19540:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
19541:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
19542:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
19543:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
19544:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
19545:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
19546:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
19547:
19548:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
19549:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
19550:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
19551:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
19552:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
19553:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
19554:
19555:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
19556:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
19557:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
19558:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
19559:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
19560:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
19561:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
19562:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
19563:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
19564:
19565:   GramsPerPound = 453.59237; // [1][7]
19566:   GramsPerDrams = GramsPerPound / 256;
19567:   GramsPerGrains = GramsPerPound / 7000;
19568:   GramsPerTons = GramsPerPound * 2000;
19569:   GramsPerLongTons = GramsPerPound * 2240;
19570:   GramsPerOunces = GramsPerPound / 16;
19571:   GramsPerStones = GramsPerPound * 14;
19572:
19573:   MaxAngle 9223372036854775808.0;
19574:   MaxTanH 5678.26170314707197474594655389854);
19575:   MaxFactorial( 1754);
19576:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
19577:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
19578:   MaxTanH( 354.89135644669199842162284618659);
19579:   MaxFactorial'LongInt'( 170);
19580:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
19581:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
19582:   MaxTanH( 44.36141955583649980270285577332 3);
19583:   MaxFactorial'LongInt'( 33);
19584:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
19585:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
19586:   PiExt( 3.14159265358979323846264338327 95);
19587:   RatioDegToRad( PiExt / 180.0);
19588:   RatioGradToRad( PiExt / 200.0);
19589:   RatioDegToGrad( 200.0 / 180.0);
19590:   RatioGradToDeg( 180.0 / 200.0);
19591:   Crc16PolynomCCITT'LongWord $1021);
19592:  Crc16PolynomIBM'LongWord $8005);
19593:  Crc16Bits'LongInt'( 16);
19594:  Crc16Bytes'LongInt'( 2);
19595:  Crc16HighBit'LongWord $8000);
19596:  NotCrc16HighBit','LongWord $7FFF);
19597:   Crc32PolynomIEEE','LongWord $04C11DB7);
19598:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
19599:  Crc32Koopman','LongWord $741B8CD7);
19600:  Crc32Bits','LongInt'( 32);
19601:  Crc32Bytes','LongInt'( 4);
19602:  Crc32HighBit','LongWord $80000000);
19603:  NotCrc32HighBit','LongWord $7FFFFFFF);
19604:
19605:   MinByte       = Low(Byte);
19606:   MaxByte       = High(Byte);
19607:   MinWord       = Low(Word);
19608:   MaxWord       = High(Word);
19609:   MinShortInt   = Low(ShortInt);
19610:   MaxShortInt   = High(ShortInt);
19611:   MinSmallInt   = Low(SmallInt);
19612:   MaxSmallInt   = High(SmallInt);
19613:   MinLongWord   = LongWord(Low(LongWord));
19614:   MaxLongWord   = LongWord(High(LongWord));
19615:   MinLongInt    = LongInt(Low(LongInt));
19616:   MaxLongInt    = LongInt(High(LongInt));
19617:   MinInt64      = Int64(Low(Int64));
19618:   MaxInt64      = Int64(High(Int64));
19619:   MinInteger    = Integer(Low(Integer));
19620:   MaxInteger    = Integer(High(Integer));
19621:   MinCardinal   = Cardinal(Low(Cardinal));
19622:   MaxCardinal   = Cardinal(High(Cardinal));
19623:   MinNativeUInt = NativeUInt(Low(NativeUInt));
19624:   MaxNativeUInt = NativeUInt(High(NativeUInt));
19625:   MinNativeInt  = NativeInt(Low(NativeInt));
19626:   MaxNativeInt  = NativeInt(High(NativeInt));
19627:  Function CosH( const Z : Float) : Float;
19628:  Function SinH( const Z : Float) : Float;
```

```
19629:  Function TanH( const Z : Float) : Float;
19630:
19631:
19632:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
19633:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
19634:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
19635:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
19636:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
19637:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
19638:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
19639:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
19640:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
19641:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
19642:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
19643:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
19644:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
19645:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
19646:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
19647:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
19648:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
19649:   MaxLog = 709.7827128933840;
19650:   MinLog = -708.3964185322641;
19651:   MaxFac = 170;
19652:   MaxGam = 171.624376956302;
19653:   MaxLgm = 2.556348E+305;
19654:   SingleCompareDelta   = 1.0E-34;
19655:   DoubleCompareDelta   = 1.0E-280;
19656:   {$IFDEF CLR}
19657:   ExtendedCompareDelta = DoubleCompareDelta;
19658:   {$ELSE}
19659:   ExtendedCompareDelta = 1.0E-4400;
19660:   {$ENDIF}
19661:   Bytes1KB  = 1024;
19662:   Bytes1MB  = 1024 * Bytes1KB;
19663:   Bytes1GB  = 1024 * Bytes1MB;
19664:   Bytes64KB = 64 * Bytes1KB;
19665:   Bytes64MB = 64 * Bytes1MB;
19666:   Bytes2GB  = 2 * LongWord(Bytes1GB);
19667:     clBlack32', $FF000000 ));
19668:     clDimGray32', $FF3F3F3F ));
19669:     clGray32', $FF7F7F7F ));
19670:     clLightGray32', $FFBFBFBF ));
19671:     clWhite32', $FFFFFFFF ));
19672:     clMaroon32', $FF7F0000 ));
19673:     clGreen32', $FF007F00 ));
19674:     clOlive32', $FF7F7F00 ));
19675:     clNavy32', $FF00007F ));
19676:     clPurple32', $FF7F007F ));
19677:     clTeal32', $FF007F7F ));
19678:     clRed32', $FFFF0000 ));
19679:     clLime32', $FF00FF00 ));
19680:     clYellow32', $FFFFFF00 ));
19681:     clBlue32', $FF0000FF ));
19682:     clFuchsia32', $FFFF00FF ));
19683:     clAqua32', $FF00FFFF ));
19684:     clAliceBlue32', $FFF0F8FF ));
19685:     clAntiqueWhite32', $FFFAEBD7 ));
19686:     clAquamarine32', $FF7FFFD4 ));
19687:     clAzure32', $FFF0FFFF ));
19688:     clBeige32', $FFF5F5DC ));
19689:     clBisque32', $FFFFE4C4 ));
19690:     clBlancheDalmond32', $FFFFEBCD ));
19691:     clBlueViolet32', $FF8A2BE2 ));
19692:     clBrown32', $FFA52A2A ));
19693:     clBurlyWood32', $FFDEB887 ));
19694:     clCadetblue32', $FF5F9EA0 ));
19695:     clChartReuse32', $FF7FFF00 ));
19696:     clChocolate32', $FFD2691E ));
19697:     clCoral32', $FFFF7F50 ));
19698:     clCornFlowerBlue32', $FF6495ED ));
19699:     clCornSilk32', $FFFFF8DC ));
19700:     clCrimson32', $FFDC143C ));
19701:     clDarkBlue32', $FF00008B ));
19702:     clDarkCyan32', $FF008B8B ));
19703:     clDarkGoldenRod32', $FFB8860B ));
19704:     clDarkGray32', $FFA9A9A9 ));
19705:     clDarkGreen32', $FF006400 ));
19706:     clDarkGrey32', $FFA9A9A9 ));
19707:     clDarkKhaki32', $FFBDB76B ));
19708:     clDarkMagenta32', $FF8B008B ));
19709:     clDarkOliveGreen32', $FF556B2F ));
19710:     clDarkOrange32', $FFFF8C00 ));
19711:     clDarkOrchid32', $FF9932CC ));
19712:     clDarkRed32', $FF8B0000 ));
19713:     clDarkSalmon32', $FFE9967A ));
19714:     clDarkSeaGreen32', $FF8FBC8F ));
19715:     clDarkSlateBlue32', $FF483D8B ));
19716:     clDarkSlateGray32', $FF2F4F4F ));
19717:     clDarkSlateGrey32', $FF2F4F4F ));
```

```
19718:     clDarkTurquoise32', $FF00CED1 ));
19719:     clDarkViolet32', $FF9400D3 ));
19720:     clDeepPink32', $FFFF1493 ));
19721:     clDeepSkyBlue32', $FF00BFFF ));
19722:     clDodgerBlue32', $FF1E90FF ));
19723:     clFireBrick32', $FFB22222 ));
19724:     clFloralWhite32', $FFFFFAF0 ));
19725:     clGainsBoro32', $FFDCDCDC ));
19726:     clGhostWhite32', $FFF8F8FF ));
19727:     clGold32', $FFFFD700 ));
19728:     clGoldenRod32', $FFDAA520 ));
19729:     clGreenYellow32', $FFADFF2F ));
19730:     clGrey32', $FF808080 ));
19731:     clHoneyDew32', $FFF0FFF0 ));
19732:     clHotPink32', $FFFF69B4 ));
19733:     clIndianRed32', $FFCD5C5C ));
19734:     clIndigo32', $FF4B0082 ));
19735:     clIvory32', $FFFFFFF0 ));
19736:     clKhaki32', $FFF0E68C ));
19737:     clLavender32', $FFE6E6FA ));
19738:     clLavenderBlush32', $FFFFF0F5 ));
19739:     clLawnGreen32', $FF7CFC00 ));
19740:     clLemonChiffon32', $FFFFFACD ));
19741:     clLightBlue32', $FFADD8E6 ));
19742:     clLightCoral32', $FFF08080 ));
19743:     clLightCyan32', $FFE0FFFF ));
19744:     clLightGoldenRodYellow32', $FFFAFAD2 ));
19745:     clLightGreen32', $FF90EE90 ));
19746:     clLightGrey32', $FFD3D3D3 ));
19747:     clLightPink32', $FFFFB6C1 ));
19748:     clLightSalmon32', $FFFFA07A ));
19749:     clLightSeagreen32', $FF20B2AA ));
19750:     clLightSkyblue32', $FF87CEFA ));
19751:     clLightSlategray32', $FF778899 ));
19752:     clLightSlategrey32', $FF778899 ));
19753:     clLightSteelblue32', $FFB0C4DE ));
19754:     clLightYellow32', $FFFFFFE0 ));
19755:     clLtGray32', $FFC0C0C0 ));
19756:     clMedGray32', $FFA0A0A4 ));
19757:     clDkGray32', $FF808080 ));
19758:     clMoneyGreen32', $FFC0DCC0 ));
19759:     clLegacySkyBlue32', $FFA6CAF0 ));
19760:     clCream32', $FFFFFBF0 ));
19761:     clLimeGreen32', $FF32CD32 ));
19762:     clLinen32', $FFFAF0E6 ));
19763:     clMediumAquamarine32', $FF66CDAA ));
19764:     clMediumBlue32', $FF0000CD ));
19765:     clMediumOrchid32', $FFBA55D3 ));
19766:     clMediumPurple32', $FF9370DB ));
19767:     clMediumSeaGreen32', $FF3CB371 ));
19768:     clMediumSlateBlue32', $FF7B68EE ));
19769:     clMediumSpringGreen32', $FF00FA9A ));
19770:     clMediumTurquoise32', $FF48D1CC ));
19771:     clMediumVioletRed32', $FFC71585 ));
19772:     clMidnightBlue32', $FF191970 ));
19773:     clMintCream32', $FFF5FFFA ));
19774:     clMistyRose32', $FFFFE4E1 ));
19775:     clMoccasin32', $FFFFE4B5 ));
19776:     clNavajoWhite32', $FFFFDEAD ));
19777:     clOldLace32', $FFFDF5E6 ));
19778:     clOliveDrab32', $FF6B8E23 ));
19779:     clOrange32', $FFFFA500 ));
19780:     clOrangeRed32', $FFFF4500 ));
19781:     clOrchid32', $FFDA70D6 ));
19782:     clPaleGoldenRod32', $FFEEE8AA ));
19783:     clPaleGreen32', $FF98FB98 ));
19784:     clPaleTurquoise32', $FFAFEEEE ));
19785:     clPaleVioletred32', $FFDB7093 ));
19786:     clPapayaWhip32', $FFFFEFD5 ));
19787:     clPeachPuff32', $FFFFDAB9 ));
19788:     clPeru32', $FFCD853F ));
19789:     clPlum32', $FFDDA0DD ));
19790:     clPowderBlue32', $FFB0E0E6 ));
19791:     clRosyBrown32', $FFBC8F8F ));
19792:     clRoyalBlue32', $FF4169E1 ));
19793:     clSaddleBrown32', $FF8B4513 ));
19794:     clSalmon32', $FFFA8072 ));
19795:     clSandyBrown32', $FFF4A460 ));
19796:     clSeaGreen32', $FF2E8B57 ));
19797:     clSeaShell32', $FFFFF5EE ));
19798:     clSienna32', $FFA0522D ));
19799:     clSilver32', $FFC0C0C0 ));
19800:     clSkyblue32', $FF87CEEB ));
19801:     clSlateBlue32', $FF6A5ACD ));
19802:     clSlateGray32', $FF708090 ));
19803:     clSlateGrey32', $FF708090 ));
19804:     clSnow32', $FFFFFAFA ));
19805:     clSpringgreen32', $FF00FF7F ));
19806:     clSteelblue32', $FF4682B4 ));
```

```
19807:     clTan32', $FFD2B48C ));
19808:     clThistle32', $FFD8BFD8 ));
19809:     clTomato32', $FFFF6347 ));
19810:     clTurquoise32', $FF40E0D0 ));
19811:     clViolet32', $FFEE82EE ));
19812:     clWheat32', $FFF5DEB3 ));
19813:     clWhitesmoke32', $FFF5F5F5 ));
19814:     clYellowgreen32', $FF9ACD32 ));
19815:     clTrWhite32', $7FFFFFFF ));
19816:     clTrBlack32', $7F000000 ));
19817:     clTrRed32', $7FFF0000 ));
19818:     clTrGreen32', $7F00FF00 ));
19819:     clTrBlue32', $7F0000FF ));
19820:   // Fixed point math constants
19821:   FixedOne = $10000;  FixedHalf = $7FFF;
19822:   FixedPI  = Round(PI * FixedOne);
19823:   FixedToFloat = 1/FixedOne;
19824:
19825:  Special Types
19826: ***************************************************
19827:   type Complex = record
19828:     X, Y : Float;
19829:   end;
19830:   type TVector     = array of Float;
19831:   TIntVector  = array of Integer;
19832:   TCompVector = array of Complex;
19833:   TBoolVector = array of Boolean;
19834:   TStrVector  = array of String;
19835:   TMatrix     = array of TVector;
19836:   TIntMatrix  = array of TIntVector;
19837:   TCompMatrix = array of TCompVector;
19838:   TBoolMatrix = array of TBoolVector;
19839:   TStrMatrix  = array of TStrVector;
19840:   TByteArray = array[0..32767] of byte; !
19841:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
19842:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
19843:   T2StringArray = array of array of string;
19844:   T2IntegerArray = array of array of integer;
19845:   AddTypeS('INT_PTR', 'Integer
19846:   AddTypeS('LONG_PTR', 'Integer
19847:   AddTypeS('UINT_PTR', 'Cardinal
19848:   AddTypeS('ULONG_PTR', 'Cardinal
19849:   AddTypeS('DWORD_PTR', 'ULONG_PTR
19850:   TIntegerDynArray', 'array of Integer
19851:   TCardinalDynArray', 'array of Cardinal
19852:   TWordDynArray', 'array of Word
19853:   TSmallIntDynArray', 'array of SmallInt
19854:   TByteDynArray', 'array of Byte
19855:   TShortIntDynArray', 'array of ShortInt
19856:   TInt64DynArray', 'array of Int64
19857:   TLongWordDynArray', 'array of LongWord
19858:   TSingleDynArray', 'array of Single
19859:   TDoubleDynArray', 'array of Double
19860:   TBooleanDynArray', 'array of Boolean
19861:   TStringDynArray', 'array of string
19862:   TWideStringDynArray', 'array of WideString
19863:   TDynByteArray     = array of Byte;
19864:   TDynShortintArray = array of Shortint;
19865:   TDynSmallintArray = array of Smallint;
19866:   TDynWordArray     = array of Word;
19867:   TDynIntegerArray  = array of Integer;
19868:   TDynLongintArray  = array of Longint;
19869:   TDynCardinalArray = array of Cardinal;
19870:   TDynInt64Array    = array of Int64;
19871:   TDynExtendedArray = array of Extended;
19872:   TDynDoubleArray   = array of Double;
19873:   TDynSingleArray   = array of Single;
19874:   TDynFloatArray    = array of Float;
19875:   TDynPointerArray  = array of Pointer;
19876:   TDynStringArray   = array of string;
19877:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
19878:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
19879:   TSynSearchOptions = set of TSynSearchOption;
19880:
19881:
19882: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
19883: ---------------------------------------------------------------------------
19884: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
19885: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
19886: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
19887: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19888: function CheckStringSum(vstring: string): integer;
19889: function HexToInt(HexNum: string): LongInt;
19890: function IntToBin(Int: Integer): String;
19891: function BinToInt(Binary: String): Integer;
19892: function HexToBin(HexNum: string): string; external2
19893: function BinToHex(Binary: String): string;
19894: function IntToFloat(i: Integer): double;
19895: function AddThousandSeparator(S: string; myChr: Char): string;
```

```
19896: function Max3(const X,Y,Z: Integer): Integer;
19897: procedure Swap(var X,Y: char); // faster without inline
19898: procedure ReverseString(var S: String);
19899: function CharToHexStr(Value: Char): string;
19900: function CharToUniCode(Value: Char): string;
19901: function Hex2Dec(Value: Str002): Byte;
19902: function HexStrCodeToStr(Value: string): string;
19903: function HexToStr(i: integer; value: string): string;
19904: function UniCodeToStr(Value: string): string;
19905: function CRC16(statement: string): string;
19906: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
19907: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
19908: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19909: Procedure ExecuteCommand(executeFile, paramstring: string);
19910: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
19911: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
19912: procedure SearchAndOpenDoc(vfilenamepath: string);
19913: procedure ShowInterfaces(myFile: string);
19914: function Fact2(av: integer): extended;
19915: Function BoolToStr(B: Boolean): string;
19916: Function GCD(x, y : LongInt) : LongInt;
19917: function LCM(m,n: longint): longint;
19918: function GetASCII: string;
19919: function GetItemHeight(Font: TFont): Integer;
19920: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
19921: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
19922: function getHINSTANCE: longword;
19923: function getHMODULE: longword;
19924: function GetASCII: string;
19925: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
19926: function WordIsOk(const AWord: string; var VW: Word): boolean;
19927: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
19928: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
19929: function SafeStr(const s: string): string;
19930: function ExtractUrlPath(const FileName: string): string;
19931: function ExtractUrlName(const FileName: string): string;
19932: function IsInternet: boolean;
19933: function RotateLeft1Bit_u32( Value: uint32): uint32;
19934: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats: Bool);
19935: procedure getEnvironmentInfo;
19936: procedure AntiFreeze;
19937: function GetCPUSpeed: Double;
19938: function IsVirtualPcGuest : Boolean;
19939: function IsVmWareGuest : Boolean;
19940: procedure StartSerialDialog;
19941: function IsWoW64: boolean;
19942: function IsWow64String(var s: string): Boolean;
19943: procedure StartThreadDemo;
19944: Function RGB(R,G,B: Byte): TColor;
19945: Function Sendln(amess: string): boolean;
19946: Procedure maXbox;
19947: Function AspectRatio(aWidth, aHeight: Integer): String;
19948: function wget(aURL, afile: string): boolean;
19949: procedure PrintList(Value: TStringList);
19950: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
19951: procedure getEnvironmentInfo;
19952: procedure AntiFreeze;
19953: function getBitmap(apath: string): TBitmap;
19954: procedure ShowMessageBig(const aText : string);
19955: function YesNoDialog(const ACaption, AMsg: string): boolean;
19956: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
19957: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19958: //function myStrToBytes(const Value: String): TBytes;
19959: //function myBytesToStr(const Value: TBytes): String;
19960: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19961: function getBitmap(apath: string): TBitmap;
19962: procedure ShowMessageBig(const aText : string);
19963: Function StrToBytes(const Value: String): TBytes;
19964: Function BytesToStr(const Value: TBytes): String;
19965: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19966: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
19967: function FindInPaths(const fileName, paths : String) : String;
19968: procedure initHexArray(var hexn: THexArray);
19969: function josephusG(n,k: integer; var graphout: string): integer;
19970: function isPowerof2(num: int64): boolean;
19971: function powerOf2(exponent: integer): int64;
19972: function getBigPI: string;
19973: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
19974:  function GetASCIILine: string;
19975: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
19976:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
19977: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19978: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19979: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
19980: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
19981: function isKeypressed: boolean;
19982: function Keypress: boolean;
```

```
19983: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19984: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
19985: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
19986: function GetOSName: string;
19987: function GetOSVersion: string;
19988: function GetOSNumber: string;
19989: function getEnvironmentString: string;
19990: procedure StrReplace(var Str: String; Old, New: String);
19991: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19992: function getTeamViewerID: string;
19993: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
19994: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
19995: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19996: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
19997: function StartSocketService: Boolean;
19998: procedure StartSocketServiceForm;
19999: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
20000: function GetFileList1(apath: string): TStringlist;
20001: procedure LetFileList(FileList: TStringlist; apath: string);
20002: procedure StartWeb(aurl: string);
20003: function GetTodayFiles(startdir, amask: string): TStringlist;
20004: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
20005: function JavahashCode(val: string): Integer;
20006: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
20007: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
20008: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
20009: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
20010: Procedure ConvertToGray(Cnv: TCanvas);
20011: function GetFileDate(aFile:string; aWithTime:Boolean):string;
20012: procedure ShowMemory;
20013: function ShowMemory2: string;
20014: function getHostIP: string;
20015: procedure ShowBitmap(bmap: TBitmap);
20016: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
20017: function CreateDBGridForm(dblist: TStringList): TListbox;
20018: function isService: boolean;
20019: function isApplication: boolean;
20020: function isTerminalSession: boolean;
20021:
20022:
20023: // News of 3.9.8 up
20024: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20025: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20026: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20027: JvChart - TJvChart Component - 2009 Public
20028: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
20029: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
20030: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
20031: DMath DLL included incl. Demos
20032: Interface Navigator menu/View/Intf Navigator
20033: Unit Explorer menu/Debug/Units Explorer
20034: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
20035: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
20036: Script History to 9 Files WebServer light /Options/Addons/WebServer
20037: Full Text Finder, JVSimLogic Simulator Package
20038: Halt-Stop Program in Menu, WebServer2, Stop Event ,
20039: Conversion Routines, Prebuild Forms, CodeSearch
20040: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20041: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20042: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20043: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
20044: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
20045: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
20046: IDE Reflection API, Session Service Shell S3
20047: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
20048: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
20049: arduino map() function, PMRandom Generator
20050: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
20051: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
20052: REST Test Lib, Multilang Component, Forth Interpreter
20053: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
20054: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
20055: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20056: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20057: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
20058: QRCode Service, add more CFunctions like CDateTime of Synapse
20059: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
20060: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
20061: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
20062: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
20063: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
20064: BOLD Package, Indy Package5, maTRIx. MATHEMAX
20065: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
20066: emax layers: system-package-component-unit-class-function-block
20067: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
20068: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
20069: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
20070: OpenGL Game Demo: ..Options/Add Ons/Reversi
20071: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
```

```
20072: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
20073: 7% performance gain (hot spot profiling)
20074: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
20075: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
20076: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
20077: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
20078:
20079: add routines in 3.9.7.5
20080: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
20081: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
20082: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
20083: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
20084: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
20085: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
20086: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
20087:
20088: ////////////////////////// TestUnits //////////////////////////
20089:   SelftestPEM;
20090:   SelfTestCFundamentUtils;
20091:   SelfTestCFileUtils;
20092:   SelfTestCDateTime;
20093:   SelfTestCTimer;
20094:   SelfTestCRandom;
20095:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
20096:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
20097:
20098:   // Note: There's no need for installing a client certificate in the
20099:   //       webbrowser. The server asks the webbrowser to send a certificate but
20100:   //       if nothing is installed the software will work because the server
20101:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
20102:   //       file: c_cacert.p12  password: c_cakey
20103:
20104:   TGraphicControl = class(TControl)
20105:   private
20106:     FCanvas: TCanvas;
20107:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20108:   protected
20109:     procedure Paint; virtual;
20110:     property Canvas: TCanvas read FCanvas;
20111:   public
20112:     constructor Create(AOwner: TComponent); override;
20113:     destructor Destroy; override;
20114:   end;
20115:
20116:   TCustomControl = class(TWinControl)
20117:   private
20118:     FCanvas: TCanvas;
20119:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20120:   protected
20121:     procedure Paint; virtual;
20122:     procedure PaintWindow(DC: HDC); override;
20123:     property Canvas: TCanvas read FCanvas;
20124:   public
20125:     constructor Create(AOwner: TComponent); override;
20126:     destructor Destroy; override;
20127:   end;
20128:     RegisterPublishedProperties;
20129:     ('ONCHANGE', 'TNotifyEvent', iptrw);
20130:     ('ONCLICK', 'TNotifyEvent', iptrw);
20131:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
20132:     ('ONENTER', 'TNotifyEvent', iptrw);
20133:     ('ONEXIT', 'TNotifyEvent', iptrw);
20134:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
20135:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
20136:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
20137:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
20138:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
20139: //*************************************************************************
20140:   // To stop the while loop, click on Options/Show Include (boolean switch)!
20141:   Control a loop in a script with a form event:
20142:   IncludeON;   //control the while loop
20143:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
20144:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
20145:
20146: //-------------------------------------------------------------------------
20147: //*************mX4 ini-file Configuration************************************
20148: //-------------------------------------------------------------------------
20149:   using config file maxboxdef.ini       menu/Help/Config File
20150:
20151: //*** Definitions for maXbox mX3 ***
20152: [FORM]
20153: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
20154: FONTSIZE=14
20155: EXTENSION=txt
20156: SCREENX=1386
20157: SCREENY=1077
20158: MEMHEIGHT=350
20159: PRINTFONT=Courier New //GUI Settings
20160: LINENUMBERS=Y   //line numbers at gutter in editor at left side
```

```
20161: EXCEPTIONLOG=Y  //store excepts + success in 2 log files see below! –menu Debug/Show Last Exceptions
20162: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
20163: BOOTSCRIPT=Y    //enabling load a boot script
20164: MEMORYREPORT=Y  //shows memory report on closing maXbox
20165: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
20166: NAVIGATOR=N     //shows function list at the right side of editor
20167: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
20168: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
20169: [WEB]
20170: IPPORT=8080     //for internal webserver – menu /Options/Add Ons/WebServer
20171: IPHOST=192.168.1.53
20172: ROOTCERT=filepathY
20173: SCERT=filepathY
20174: RSAKEY=filepathY
20175: VERSIONCHECK=Y
20176:
20177: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
20178: Also possible to set report memory in script to override ini setting
20179: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
20180:
20181: After Change the ini file you can reload the file with ../Help/Config Update
20182:
20183: //-------------------------------------------------------------------------
20184: //**************mX4 maildef.ini ini-file Configuration********************
20185: //-------------------------------------------------------------------------
20186: //*** Definitions for maXMail ***
20187: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
20188: [MAXMAIL]
20189: HOST=getmail.softwareschule.ch
20190: USER=mailusername
20191: PASS=password
20192: PORT=110
20193: SSL=Y
20194: BODY=Y
20195: LAST=5
20196:
20197: ADO Connection String:
20198: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
20199: \452_dbtreeview2access.txt
20200: program TestDbTreeViewMainForm2_ACCESS;
20201:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20202:                    +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20203:
20204: OpenSSL Lib:  unit ssl_openssl_lib;
20205: {$IFDEF CIL}
20206: const
20207:   {$IFDEF LINUX}
20208:   DLLSSLName = 'libssl.so';
20209:   DLLUtilName = 'libcrypto.so';
20210:   {$ELSE}
20211:   DLLSSLName = 'ssleay32.dll';
20212:   DLLUtilName = 'libeay32.dll';
20213:   {$ENDIF}
20214: {$ELSE}
20215: var
20216:   {$IFNDEF MSWINDOWS}
20217:     {$IFDEF DARWIN}
20218:     DLLSSLName: string = 'libssl.dylib';
20219:     DLLUtilName: string = 'libcrypto.dylib';
20220:     {$ELSE}
20221:     DLLSSLName: string = 'libssl.so';
20222:     DLLUtilName: string = 'libcrypto.so';
20223:     {$ENDIF}
20224:   {$ELSE}
20225:   DLLSSLName: string = 'ssleay32.dll';
20226:   DLLSSLName2: string = 'libssl32.dll';
20227:   DLLUtilName: string = 'libeay32.dll';
20228:   {$ENDIF}
20229: {$ENDIF}
20230:
20231:
20232: //-------------------------------------------------------------------------
20233: //**************mX4 Macro Tags  ***********************************************
20234: //---------------------------------------------------------- ----------------
20235:
20236:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
     E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
20237:
20238: //Tag Macros
20239:
20240:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20241:
20242: //Tag Macros
20243: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20244: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20245: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
20246: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20247: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20248: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
```

```
20249: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20250: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
20251: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
20252:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
20253: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
20254: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20255:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20256: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
20257:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
20258: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
20259:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
20260:
20261: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20262:
20263: //Replace Macros
20264:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20265:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20266:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20267:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
20268:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
20269:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
20270:
20271:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20272:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
20273: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20274:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
20275:
20276: //-------------------------------------------------------------------------
20277: //**************mX4 ToDo List Tags  ../Help/ToDo List************************
20278: //------------------------------------------------------- ---------------------
20279:
20280:     while I < sl.Count do begin
20281: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
20282:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
20283:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
20284:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
20285:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20286:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
20287:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20288:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
20289:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20290:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
20291:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
20292:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
20293:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
20294:       Inc(I);
20295:     end;
20296:
20297: //-------------------------------------------------------------------------
20298: //**************mX4 Public  Tools API ***************************************
20299: //-------------------------------------------------------------------------
20300:   file : unit uPSI_fMain.pas;               {$OTAP} Open Tools API Catalog
20301:  // Those functions concern the editor and preprocessor, all of the IDE
20302:  Example: Call it with maxform1.Info1Click(self)
20303:  Note: Call all Methods with maxForm1., e.g.:
20304:                       maxForm1.ShellStyle1Click(self);
20305:
20306: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20307: begin
20308:  Const('BYTECODE','String 'bytecode.txt'
20309:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
20310:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
20311:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
20312:  Const('PSINC','String PS Includes (*.inc)|*.INC
20313:  Const('DEFFILENAME','String 'firstdemo.txt
20314:  Const('DEFINIFILE','String 'maxboxdef.ini
20315:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
20316:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
20317:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
20318:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
20319:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
20320:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
20321:  Const('INCLUDEBOX','String 'pas_includebox.inc
20322:  Const('BOOTSCRIPT','String 'maxbootscript.txt
20323:  Const('MBVERSION','String '3.9.9.100
20324:  Const('VERSION','String'3.9.9.100
20325:  Const('MBVER','String '399
20326:  Const('MBVERI','Integer'(399);
20327:  Const('MBVERIALL','Integer'(399100);
20328:  Const('EXENAME','String 'maXbox3.exe
20329:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
20330:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
20331:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
20332:  Const('MXINTERNETCHECK','String 'www.ask.com
20333:  Const('MXMAIL','String 'max@kleiner.com
20334:  Const('TAB','Char #$09);
20335:  Const('CODECOMPLETION','String 'bds_delphi.dci
20336:  SIRegister_TMaxForm1(CL);
20337: end;
```

```
20338:
20339:  with FindClass('TForm'),'TMaxForm1') do begin
20340:    memo2', 'TMemo', iptrw);
20341:    memo1', 'TSynMemo', iptrw);
20342:    CB1SCList', 'TComboBox', iptrw);
20343:    mxNavigator', 'TComboBox', iptrw);
20344:    IPHost', 'string', iptrw);
20345:    IPPort', 'integer', iptrw);
20346:    COMPort', 'integer', iptrw);      //3.9.6.4
20347:    Splitter1', 'TSplitter', iptrw);
20348:    PSScript', 'TPSScript', iptrw);
20349:    PS3DllPlugin', 'TPSDllPlugin', iptrw);
20350:    MainMenu1', 'TMainMenu', iptrw);
20351:    Program1', 'TMenuItem', iptrw);
20352:    Compile1', 'TMenuItem', iptrw);
20353:    Files1', 'TMenuItem', iptrw);
20354:    open1', 'TMenuItem', iptrw);
20355:    Save2', 'TMenuItem', iptrw);
20356:    Options1', 'TMenuItem', iptrw);
20357:    Savebefore1', 'TMenuItem', iptrw);
20358:    Largefont1', 'TMenuItem', iptrw);
20359:    sBytecode1', 'TMenuItem', iptrw);
20360:    Saveas3', 'TMenuItem', iptrw);
20361:    Clear1', 'TMenuItem', iptrw);
20362:    Slinenumbers1', 'TMenuItem', iptrw);
20363:    About1', 'TMenuItem', iptrw);
20364:    Search1', 'TMenuItem', iptrw);
20365:    SynPasSyn1', 'TSynPasSyn', iptrw);
20366:    memo1', 'TSynMemo', iptrw);
20367:    SynEditSearch1', 'TSynEditSearch', iptrw);
20368:    WordWrap1', 'TMenuItem', iptrw);
20369:    XPManifest1', 'TXPManifest', iptrw);
20370:    SearchNext1', 'TMenuItem', iptrw);
20371:    Replace1', 'TMenuItem', iptrw);
20372:    PSImport_Controls1', 'TPSImport_Controls', iptrw);
20373:    PSImport_Classes1', 'TPSImport_Classes', iptrw);
20374:    ShowInclude1', 'TMenuItem', iptrw);
20375:    SynEditPrint1', 'TSynEditPrint', iptrw);
20376:    Printout1', 'TMenuItem', iptrw);
20377:    mnPrintColors1', 'TMenuItem', iptrw);
20378:    dlgFilePrint', 'TPrintDialog', iptrw);
20379:    dlgPrintFont1', 'TFontDialog', iptrw);
20380:    mnuPrintFont1', 'TMenuItem', iptrw);
20381:    Include1', 'TMenuItem', iptrw);
20382:    CodeCompletionList1', 'TMenuItem', iptrw);
20383:    IncludeList1', 'TMenuItem', iptrw);
20384:    ImageList1', 'TImageList', iptrw);
20385:    ImageList2', 'TImageList', iptrw);
20386:    CoolBar1', 'TCoolBar', iptrw);
20387:    ToolBar1', 'TToolBar', iptrw);
20388:    tbtnLoad', 'TToolButton', iptrw);
20389:    ToolButton2', 'TToolButton', iptrw);
20390:    tbtnFind', 'TToolButton', iptrw);
20391:    tbtnCompile', 'TToolButton', iptrw);
20392:    tbtnTrans', 'TToolButton', iptrw);
20393:    tbtnUseCase', 'TToolButton', iptrw);   //3.8
20394:    toolbtnTutorial', 'TToolButton', iptrw);
20395:    tbtn6res', 'TToolButton', iptrw);
20396:    ToolButton5', 'TToolButton', iptrw);
20397:    ToolButton1', 'TToolButton', iptrw);
20398:    ToolButton3', 'TToolButton', iptrw);
20399:    statusBar1', 'TStatusBar', iptrw);
20400:    SaveOutput1', 'TMenuItem', iptrw);
20401:    ExportClipboard1', 'TMenuItem', iptrw);
20402:    Close1', 'TMenuItem', iptrw);
20403:    Manual1', 'TMenuItem', iptrw);
20404:    About2', 'TMenuItem', iptrw);
20405:    loadLastfile1', 'TMenuItem', iptrw);
20406:    imglogo', 'TImage', iptrw);
20407:    cedebug', 'TPSScriptDebugger', iptrw);
20408:    debugPopupMenu1', 'TPopupMenu', iptrw);
20409:    BreakPointMenu', 'TMenuItem', iptrw);
20410:    Decompile1', 'TMenuItem', iptrw);
20411:    StepInto1', 'TMenuItem', iptrw);
20412:    StepOut1', 'TMenuItem', iptrw);
20413:    Reset1', 'TMenuItem', iptrw);
20414:    DebugRun1', 'TMenuItem', iptrw);
20415:    PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20416:    PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20417:    PSImport_Forms1', 'TPSImport_Forms', iptrw);
20418:    PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20419:    tutorial4', 'TMenuItem', iptrw);
20420:    ExporttoClipboard1', 'TMenuItem', iptrw);
20421:    ImportfromClipboard1', 'TMenuItem', iptrw);
20422:    N4', 'TMenuItem', iptrw);
20423:    N5', 'TMenuItem', iptrw);
20424:    N6', 'TMenuItem', iptrw);
20425:    ImportfromClipboard2', 'TMenuItem', iptrw);
20426:    tutorial1', 'TMenuItem', iptrw);
```

```
20427:     N7', 'TMenuItem', iptrw);
20428:     ShowSpecChars1', 'TMenuItem', iptrw);
20429:     OpenDirectory1', 'TMenuItem', iptrw);
20430:     procMess', 'TMenuItem', iptrw);
20431:     tbtnUseCase', 'TToolButton', iptrw);
20432:     ToolButton7', 'TToolButton', iptrw);
20433:     EditFont1', 'TMenuItem', iptrw);
20434:     UseCase1', 'TMenuItem', iptrw);
20435:     tutorial21', 'TMenuItem', iptrw);
20436:     OpenUseCase1', 'TMenuItem', iptrw);
20437:     PSImport_DB1', 'TPSImport_DB', iptrw);
20438:     tutorial31', 'TMenuItem', iptrw);
20439:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20440:     HTMLSyntax1', 'TMenuItem', iptrw);
20441:     ShowInterfaces1', 'TMenuItem', iptrw);
20442:     Tutorial5', 'TMenuItem', iptrw);
20443:     AllFunctionsList1', 'TMenuItem', iptrw);
20444:     ShowLastException1', 'TMenuItem', iptrw);
20445:     PlayMP31', 'TMenuItem', iptrw);
20446:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
20447:     texSyntax1', 'TMenuItem', iptrw);
20448:     N8', 'TMenuItem', iptrw);
20449:     GetEMails1', 'TMenuItem', iptrw);
20450:     SynCppSyn1', 'TSynCppSyn', iptrw);
20451:     CSyntax1', 'TMenuItem', iptrw);
20452:     Tutorial6', 'TMenuItem', iptrw);
20453:     New1', 'TMenuItem', iptrw);
20454:     AllObjectsList1', 'TMenuItem', iptrw);
20455:     LoadBytecode1', 'TMenuItem', iptrw);
20456:     CipherFile1', 'TMenuItem', iptrw);
20457:     N9', 'TMenuItem', iptrw);
20458:     N10', 'TMenuItem', iptrw);
20459:     Tutorial11', 'TMenuItem', iptrw);
20460:     Tutorial71', 'TMenuItem', iptrw);
20461:     UpdateService1', 'TMenuItem', iptrw);
20462:     PascalSchool1', 'TMenuItem', iptrw);
20463:     Tutorial81', 'TMenuItem', iptrw);
20464:     DelphiSite1', 'TMenuItem', iptrw);
20465:     Output1', 'TMenuItem', iptrw);
20466:     TerminalStyle1', 'TMenuItem', iptrw);
20467:     ReadOnly1', 'TMenuItem', iptrw);
20468:     ShellStyle1', 'TMenuItem', iptrw);
20469:     BigScreen1', 'TMenuItem', iptrw);
20470:     Tutorial91', 'TMenuItem', iptrw);
20471:     SaveOutput2', 'TMenuItem', iptrw);
20472:     N11', 'TMenuItem', iptrw);
20473:     SaveScreenshot', 'TMenuItem', iptrw);
20474:     Tutorial101', 'TMenuItem', iptrw);
20475:     SQLSyntax1', 'TMenuItem', iptrw);
20476:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
20477:     Console1', 'TMenuItem', iptrw);
20478:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
20479:     XMLSyntax1', 'TMenuItem', iptrw);
20480:     ComponentCount1', 'TMenuItem', iptrw);
20481:     NewInstance1', 'TMenuItem', iptrw);
20482:     toolbtnTutorial', 'TToolButton', iptrw);
20483:     Memory1', 'TMenuItem', iptrw);
20484:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
20485:     JavaSyntax1', 'TMenuItem', iptrw);
20486:     SyntaxCheck1', 'TMenuItem', iptrw);
20487:     Tutorial10Statistics1', 'TMenuItem', iptrw);
20488:     ScriptExplorer1', 'TMenuItem', iptrw);
20489:     FormOutput1', 'TMenuItem', iptrw);
20490:     ArduinoDump1', 'TMenuItem', iptrw);
20491:     AndroidDump1', 'TMenuItem', iptrw);
20492:     GotoEnd1', 'TMenuItem', iptrw);
20493:     AllResourceList1', 'TMenuItem', iptrw);
20494:     ToolButton4', 'TToolButton', iptrw);
20495:     tbtn6res', 'TToolButton', iptrw);
20496:     Tutorial11Forms1', 'TMenuItem', iptrw);
20497:     Tutorial12SQL1', 'TMenuItem', iptrw);
20498:     ResourceExplore1', 'TMenuItem', iptrw);
20499:     Info1', 'TMenuItem', iptrw);
20500:     N12', 'TMenuItem', iptrw);
20501:     CryptoBox1', 'TMenuItem', iptrw);
20502:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
20503:     CipherFile2', 'TMenuItem', iptrw);
20504:     N13', 'TMenuItem', iptrw);
20505:     ModulesCount1', 'TMenuItem', iptrw);
20506:     AddOns2', 'TMenuItem', iptrw);
20507:     N4GewinntGame1', 'TMenuItem', iptrw);
20508:     DocuforAddOns1', 'TMenuItem', iptrw);
20509:     Tutorial14Async1', 'TMenuItem', iptrw);
20510:     Lessons15Review1', 'TMenuItem', iptrw);
20511:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
20512:     PHPSyntax1', 'TMenuItem', iptrw);
20513:     Breakpoint1', 'TMenuItem', iptrw);
20514:     SerialRS2321', 'TMenuItem', iptrw);
20515:     N14', 'TMenuItem', iptrw);
```

```
20516:     SynCSSyn1', 'TSynCSSyn', iptrw);
20517:     CSyntax2', 'TMenuItem', iptrw);
20518:     Calculator1', 'TMenuItem', iptrw);
20519:     tbtnSerial', 'TToolButton', iptrw);
20520:     ToolButton8', 'TToolButton', iptrw);
20521:     Tutorial151', 'TMenuItem', iptrw);
20522:     N15', 'TMenuItem', iptrw);
20523:     N16', 'TMenuItem', iptrw);
20524:     ControlBar1', 'TControlBar', iptrw);
20525:     ToolBar2', 'TToolBar', iptrw);
20526:     BtnOpen', 'TToolButton', iptrw);
20527:     BtnSave', 'TToolButton', iptrw);
20528:     BtnPrint', 'TToolButton', iptrw);
20529:     BtnColors', 'TToolButton', iptrw);
20530:     btnClassReport', 'TToolButton', iptrw);
20531:     BtnRotateRight', 'TToolButton', iptrw);
20532:     BtnFullSize', 'TToolButton', iptrw);
20533:     BtnFitToWindowSize', 'TToolButton', iptrw);
20534:     BtnZoomMinus', 'TToolButton', iptrw);
20535:     BtnZoomPlus', 'TToolButton', iptrw);
20536:     Panel1', 'TPanel', iptrw);
20537:     LabelBrettgroesse', 'TLabel', iptrw);
20538:     CB1SCList', 'TComboBox', iptrw);
20539:     ImageListNormal', 'TImageList', iptrw);
20540:     spbtnexplore', 'TSpeedButton', iptrw);
20541:     spbtnexample', 'TSpeedButton', iptrw);
20542:     spbsaveas', 'TSpeedButton', iptrw);
20543:     imglogobox', 'TImage', iptrw);
20544:     EnlargeFont1', 'TMenuItem', iptrw);
20545:     EnlargeFont2', 'TMenuItem', iptrw);
20546:     ShrinkFont1', 'TMenuItem', iptrw);
20547:     ThreadDemo1', 'TMenuItem', iptrw);
20548:     HEXEditor1', 'TMenuItem', iptrw);
20549:     HEXView1', 'TMenuItem', iptrw);
20550:     HEXInspect1', 'TMenuItem', iptrw);
20551:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
20552:     ExporttoHTML1', 'TMenuItem', iptrw);
20553:     ClassCount1', 'TMenuItem', iptrw);
20554:     HTMLOutput1', 'TMenuItem', iptrw);
20555:     HEXEditor2', 'TMenuItem', iptrw);
20556:     Minesweeper1', 'TMenuItem', iptrw);
20557:     N17', 'TMenuItem', iptrw);
20558:     PicturePuzzle1', 'TMenuItem', iptrw);
20559:     sbvclhelp', 'TSpeedButton', iptrw);
20560:     DependencyWalker1', 'TMenuItem', iptrw);
20561:     WebScanner1', 'TMenuItem', iptrw);
20562:     View1', 'TMenuItem', iptrw);
20563:     mnToolbar1', 'TMenuItem', iptrw);
20564:     mnStatusbar2', 'TMenuItem', iptrw);
20565:     mnConsole2', 'TMenuItem', iptrw);
20566:     mnCoolbar2', 'TMenuItem', iptrw);
20567:     mnSplitter2', 'TMenuItem', iptrw);
20568:     WebServer1', 'TMenuItem', iptrw);
20569:     Tutorial17Server1', 'TMenuItem', iptrw);
20570:     Tutorial18Arduino1', 'TMenuItem', iptrw);
20571:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
20572:     PerlSyntax1', 'TMenuItem', iptrw);
20573:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
20574:     PythonSyntax1', 'TMenuItem', iptrw);
20575:     DMathLibrary1', 'TMenuItem', iptrw);
20576:     IntfNavigator1', 'TMenuItem', iptrw);
20577:     EnlargeFontConsole1', 'TMenuItem', iptrw);
20578:     ShrinkFontConsole1', 'TMenuItem', iptrw);
20579:     SetInterfaceList1', 'TMenuItem', iptrw);
20580:     popintfList', 'TPopupMenu', iptrw);
20581:     intfAdd1', 'TMenuItem', iptrw);
20582:     intfDelete1', 'TMenuItem', iptrw);
20583:     intfRefactor1', 'TMenuItem', iptrw);
20584:     Defactor1', 'TMenuItem', iptrw);
20585:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
20586:     Tutorial20Regex', 'TMenuItem', iptrw);
20587:     N18', 'TMenuItem', iptrw);
20588:     ManualE1', 'TMenuItem', iptrw);
20589:     FullTextFinder1', 'TMenuItem', iptrw);
20590:     Move1', 'TMenuItem', iptrw);
20591:     FractalDemo1', 'TMenuItem', iptrw);
20592:     Tutorial21Android1', 'TMenuItem', iptrw);
20593:     Tutorial0Function1', 'TMenuItem', iptrw);
20594:     SimuLogBox1', 'TMenuItem', iptrw);
20595:     OpenExamples1', 'TMenuItem', iptrw);
20596:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
20597:     JavaScriptSyntax1', 'TMenuItem', iptrw);
20598:     Halt1', 'TMenuItem', iptrw);
20599:     CodeSearch1', 'TMenuItem', iptrw);
20600:     SynRubySyn1', 'TSynRubySyn', iptrw);
20601:     RubySyntax1', 'TMenuItem', iptrw);
20602:     Undo1', 'TMenuItem', iptrw);
20603:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
20604:     LinuxShellScript1', 'TMenuItem', iptrw);
```

```
20605:     Rename1', 'TMenuItem', iptrw);
20606:     spdcodesearch', 'TSpeedButton', iptrw);
20607:     Preview1', 'TMenuItem', iptrw);
20608:     Tutorial22Services1', 'TMenuItem', iptrw);
20609:     Tutorial23RealTime1', 'TMenuItem', iptrw);
20610:     Configuration1', 'TMenuItem', iptrw);
20611:     MP3Player1', 'TMenuItem', iptrw);
20612:     DLLSpy1', 'TMenuItem', iptrw);
20613:     SynURIOpener1', 'TSynURIOpener', iptrw);
20614:     SynURISyn1', 'TSynURISyn', iptrw);
20615:     URILinksClicks1', 'TMenuItem', iptrw);
20616:     EditReplace1', 'TMenuItem', iptrw);
20617:     GotoLine1', 'TMenuItem', iptrw);
20618:     ActiveLineColor1', 'TMenuItem', iptrw);
20619:     ConfigFile1', 'TMenuItem', iptrw);
20620:     Sort1Intflist', 'TMenuItem', iptrw);
20621:     Redo1', 'TMenuItem', iptrw);
20622:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
20623:     Tutorial25Configuration1', 'TMenuItem', iptrw);
20624:     IndentSelection1', 'TMenuItem', iptrw);
20625:     UnindentSection1', 'TMenuItem', iptrw);
20626:     SkyStyle1', 'TMenuItem', iptrw);
20627:     N19', 'TMenuItem', iptrw);
20628:     CountWords1', 'TMenuItem', iptrw);
20629:     imbookmarkimages', 'TImageList', iptrw);
20630:     Bookmark11', 'TMenuItem', iptrw);
20631:     N20', 'TMenuItem', iptrw);
20632:     Bookmark21', 'TMenuItem', iptrw);
20633:     Bookmark31', 'TMenuItem', iptrw);
20634:     Bookmark41', 'TMenuItem', iptrw);
20635:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
20636:
20637:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
20638:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
20639:     Procedure PSScriptCompile( Sender : TPSScript)
20640:     Procedure Compile1Click( Sender : TObject)
20641:     Procedure PSScriptExecute( Sender : TPSScript)
20642:     Procedure open1Click( Sender : TObject)
20643:     Procedure Save2Click( Sender : TObject)
20644:     Procedure Savebefore1Click( Sender : TObject)
20645:     Procedure Largefont1Click( Sender : TObject)
20646:     Procedure FormActivate( Sender : TObject)
20647:     Procedure SBytecode1Click( Sender : TObject)
20648:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
20649:     Procedure Saveas3Click( Sender : TObject)
20650:     Procedure Clear1Click( Sender : TObject)
20651:     Procedure Slinenumbers1Click( Sender : TObject)
20652:     Procedure About1Click( Sender : TObject)
20653:     Procedure Search1Click( Sender : TObject)
20654:     Procedure FormCreate( Sender : TObject)
20655:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
20656:                                                     var Action : TSynReplaceAction)
20657:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
20658:     Procedure WordWrap1Click( Sender : TObject)
20659:     Procedure SearchNext1Click( Sender : TObject)
20660:     Procedure Replace1Click( Sender : TObject)
20661:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
20662:     Procedure ShowInclude1Click( Sender : TObject)
20663:     Procedure Printout1Click( Sender : TObject)
20664:     Procedure mnuPrintFont1Click( Sender : TObject)
20665:     Procedure Include1Click( Sender : TObject)
20666:     Procedure FormDestroy( Sender : TObject)
20667:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
20668:     Procedure UpdateView1Click( Sender : TObject)
20669:     Procedure CodeCompletionList1Click( Sender : TObject)
20670:     Procedure SaveOutput1Click( Sender : TObject)
20671:     Procedure ExportClipboard1Click( Sender : TObject)
20672:     Procedure Close1Click( Sender : TObject)
20673:     Procedure Manual1Click( Sender : TObject)
20674:     Procedure LoadLastFile1Click( Sender : TObject)
20675:     Procedure Memo1Change( Sender : TObject)
20676:     Procedure Decompile1Click( Sender : TObject)
20677:     Procedure StepInto1Click( Sender : TObject)
20678:     Procedure StepOut1Click( Sender : TObject)
20679:     Procedure Reset1Click( Sender : TObject)
20680:     Procedure cedebugAfterExecute( Sender : TPSScript)
20681:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
20682:     Procedure cedebugCompile( Sender : TPSScript)
20683:     Procedure cedebugExecute( Sender : TPSScript)
20684:     Procedure cedebugIdle( Sender : TObject)
20685:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
20686:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
20687:     Procedure BreakPointMenuClick( Sender : TObject)
20688:     Procedure DebugRun1Click( Sender : TObject)
20689:     Procedure tutorial4Click( Sender : TObject)
20690:     Procedure ImportfromClipboard1Click( Sender : TObject)
20691:     Procedure ImportfromClipboard2Click( Sender : TObject)
20692:     Procedure tutorial1Click( Sender : TObject)
20693:     Procedure ShowSpecChars1Click( Sender : TObject)
```

```
20694:     Procedure StatusBar1DblClick( Sender : TObject)
20695:     Procedure PSScriptLine( Sender : TObject)
20696:     Procedure OpenDirectory1Click( Sender : TObject)
20697:     Procedure procMessClick( Sender : TObject)
20698:     Procedure tbtnUseCaseClick( Sender : TObject)
20699:     Procedure EditFont1Click( Sender : TObject)
20700:     Procedure tutorial21Click( Sender : TObject)
20701:     Procedure tutorial31Click( Sender : TObject)
20702:     Procedure HTMLSyntax1Click( Sender : TObject)
20703:     Procedure ShowInterfaces1Click( Sender : TObject)
20704:     Procedure Tutorial5Click( Sender : TObject)
20705:     Procedure ShowLastException1Click( Sender : TObject)
20706:     Procedure PlayMP31Click( Sender : TObject)
20707:     Procedure AllFunctionsList1Click( Sender : TObject)
20708:     Procedure texSyntax1Click( Sender : TObject)
20709:     Procedure GetEMails1Click( Sender : TObject)
20710:     procedure DelphiSite1Click(Sender: TObject);
20711:     procedure TerminalStyle1Click(Sender: TObject);
20712:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
20713:     procedure ShellStyle1Click(Sender: TObject);
20714:     procedure Console1Click(Sender: TObject);     //3.2
20715:     procedure BigScreen1Click(Sender: TObject);
20716:     procedure Tutorial91Click(Sender: TObject);
20717:     procedure SaveScreenshotClick(Sender: TObject);
20718:     procedure Tutorial101Click(Sender: TObject);
20719:     procedure SQLSyntax1Click(Sender: TObject);
20720:     procedure XMLSyntax1Click(Sender: TObject);
20721:     procedure ComponentCount1Click(Sender: TObject);
20722:     procedure NewInstance1Click(Sender: TObject);
20723:     procedure CSyntax1Click(Sender: TObject);
20724:     procedure Tutorial6Click(Sender: TObject);
20725:     procedure New1Click(Sender: TObject);
20726:     procedure AllObjectsList1Click(Sender: TObject);
20727:     procedure LoadBytecode1Click(Sender: TObject);
20728:     procedure CipherFile1Click(Sender: TObject);  //V3.5
20729:     procedure NewInstance1Click(Sender: TObject);
20730:     procedure toolbtnTutorialClick(Sender: TObject);
20731:     procedure Memory1Click(Sender: TObject);
20732:     procedure JavaSyntax1Click(Sender: TObject);
20733:     procedure SyntaxCheck1Click(Sender: TObject);
20734:     procedure ScriptExplorer1Click(Sender: TObject);
20735:     procedure FormOutput1Click(Sender: TObject);  //V3.6
20736:     procedure GotoEnd1Click(Sender: TObject);
20737:     procedure AllResourceList1Click(Sender: TObject);
20738:     procedure tbtn6resClick(Sender: TObject);   //V3.7
20739:     procedure Info1Click(Sender: TObject);
20740:     procedure Tutorial10Statistics1Click(Sender: TObject);
20741:     procedure Tutorial11Forms1Click(Sender: TObject);
20742:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
20743:     procedure ResourceExplore1Click(Sender: TObject);
20744:     procedure Info1Click(Sender: TObject);
20745:     procedure CryptoBox1Click(Sender: TObject);
20746:     procedure ModulesCount1Click(Sender: TObject);
20747:     procedure N4GewinntGame1Click(Sender: TObject);
20748:     procedure PHPSyntax1Click(Sender: TObject);
20749:     procedure SerialRS2321Click(Sender: TObject);
20750:     procedure CSyntax2Click(Sender: TObject);
20751:     procedure Calculator1Click(Sender: TObject);
20752:     procedure Tutorial13Ciphering1Click(Sender: TObject);
20753:     procedure Tutorial14Async1Click(Sender: TObject);
20754:     procedure PHPSyntax1Click(Sender: TObject);
20755:     procedure BtnZoomPlusClick(Sender: TObject);
20756:     procedure BtnZoomMinusClick(Sender: TObject);
20757:     procedure btnClassReportClick(Sender: TObject);
20758:     procedure ThreadDemo1Click(Sender: TObject);
20759:     procedure HEXView1Click(Sender: TObject);
20760:     procedure ExporttoHTML1Click(Sender: TObject);
20761:     procedure Minesweeper1Click(Sender: TObject);
20762:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
20763:     procedure sbvclhelpClick(Sender: TObject);
20764:     procedure DependencyWalker1Click(Sender: TObject);
20765:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
20766:     procedure WebScanner1Click(Sender: TObject);
20767:     procedure mnToolbar1Click(Sender: TObject);
20768:     procedure mnStatusbar2Click(Sender: TObject);
20769:     procedure mnConsole2Click(Sender: TObject);
20770:     procedure mnCoolbar2Click(Sender: TObject);
20771:     procedure mnSplitter2Click(Sender: TObject);
20772:     procedure WebServer1Click(Sender: TObject);
20773:     procedure PerlSyntax1Click(Sender: TObject);
20774:     procedure PythonSyntax1Click(Sender: TObject);
20775:     procedure DMathLibrary1Click(Sender: TObject);
20776:     procedure IntfNavigator1Click(Sender: TObject);
20777:     procedure FullTextFinder1Click(Sender: TObject);
20778:     function AppName: string;
20779:     function ScriptName: string;
20780:     function LastName: string;
20781:     procedure FractalDemo1Click(Sender: TObject);
20782:     procedure SimuLogBox1Click(Sender: TObject);
```

```
20783:     procedure OpenExamples1Click(Sender: TObject);
20784:     procedure Halt1Click(Sender: TObject);
20785:     procedure Stop;
20786:     procedure CodeSearch1Click(Sender: TObject);
20787:     procedure RubySyntax1Click(Sender: TObject);
20788:     procedure Undo1Click(Sender: TObject);
20789:     procedure LinuxShellScript1Click(Sender: TObject);
20790:     procedure WebScannerDirect(urls: string);
20791:     procedure WebScanner(urls: string);
20792:     procedure LoadInterfaceList2;
20793:     procedure DLLSpy1Click(Sender: TObject);
20794:     procedure Memo1DblClick(Sender: TObject);
20795:     procedure URILinksClicks1Click(Sender: TObject);
20796:     procedure GotoLine1Click(Sender: TObject);
20797:     procedure ConfigFile1Click(Sender: TObject);
20798:     Procedure Sort1IntflistClick( Sender : TObject)
20799:     Procedure Redo1Click( Sender : TObject)
20800:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
20801:     Procedure IndentSelection1Click( Sender : TObject)
20802:     Procedure UnindentSection1Click( Sender : TObject)
20803:     Procedure SkyStyle1Click( Sender : TObject)
20804:     Procedure CountWords1Click( Sender : TObject)
20805:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
20806:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
20807:     Procedure Bookmark11Click( Sender : TObject)
20808:     Procedure Bookmark21Click( Sender : TObject)
20809:     Procedure Bookmark31Click( Sender : TObject)
20810:     Procedure Bookmark41Click( Sender : TObject)
20811:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
20812:     'STATMemoryReport', 'boolean', iptrw);
20813:     'IPPort', 'integer', iptrw);
20814:     'COMPort', 'integer', iptrw);
20815:     'lbintflist', 'TListBox', iptrw);
20816:     Function GetStatChange : boolean
20817:     Procedure SetStatChange( vstat : boolean)
20818:     Function GetActFileName : string
20819:     Procedure SetActFileName( vname : string)
20820:     Function GetLastFileName : string
20821:     Procedure SetLastFileName( vname : string)
20822:     Procedure WebScannerDirect( urls : string)
20823:     Procedure LoadInterfaceList2
20824:     Function GetStatExecuteShell : boolean
20825:     Procedure DoEditorExecuteCommand( EditorCommand : word)
20826:     function GetActiveLineColor: TColor
20827:     procedure SetActiveLineColor(acolor: TColor)
20828:     procedure ScriptListbox1Click(Sender: TObject);
20829:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
20830:     procedure EnlargeGutter1Click(Sender: TObject);
20831:     procedure Tetris1Click(Sender: TObject);
20832:     procedure ToDoList1Click(Sender: TObject);
20833:     procedure ProcessList1Click(Sender: TObject);
20834:     procedure MetricReport1Click(Sender: TObject);
20835:     procedure ProcessList1Click(Sender: TObject);
20836:     procedure TCPSockets1Click(Sender: TObject);
20837:     procedure ConfigUpdate1Click(Sender: TObject);
20838:     procedure ADOWorkbench1Click(Sender: TObject);
20839:     procedure SocketServer1Click(Sender: TObject);
20840:     procedure FormDemo1Click(Sender: TObject);
20841:     procedure Richedit1Click(Sender: TObject);
20842:     procedure SimpleBrowser1Click(Sender: TObject);
20843:     procedure DOSShell1Click(Sender: TObject);
20844:     procedure SynExport1Click(Sender: TObject);
20845:     procedure ExporttoRTF1Click(Sender: TObject);
20846:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
20847:     procedure SOAPTester1Click(Sender: TObject);
20848:     procedure Sniffer1Click(Sender: TObject);
20849:     procedure AutoDetectSyntax1Click(Sender: TObject);
20850:     procedure FPlot1Click(Sender: TObject);
20851:     procedure PasStyle1Click(Sender: TObject);
20852:     procedure Tutorial183RGBLED1Click(Sender: TObject);
20853:     procedure Reversi1Click(Sender: TObject);
20854:     procedure ManualmaXbox1Click(Sender: TObject);
20855:     procedure BlaisePascalMagazine1Click(Sender: TObject);
20856:     procedure AddToDo1Click(Sender: TObject);
20857:     procedure CreateGUID1Click(Sender: TObject);
20858:     procedure Tutorial27XML1Click(Sender: TObject);
20859:     procedure CreateDLLStub1Click(Sender: TObject);
20860:     procedure Tutorial28DLL1Click(Sender: TObject);');
20861:     procedure ResetKeyPressed;');
20862:     procedure KeyPressedFalse;
20863:     procedure FileChanges1Click(Sender: TObject);');
20864:     procedure OpenGLTry1Click(Sender: TObject);');
20865:     procedure AllUnitList1Click(Sender: TObject);');
20866:     procedure Tutorial29UMLClick(Sender: TObject);
20867:     procedure CreateHeader1Click(Sender: TObject);
20868:     procedure Oscilloscope1Click(Sender: TObject);');
20869:     procedure Tutorial30WOT1Click(Sender: TObject);');
20870:     procedure GetWebScript1Click(Sender: TObject);');
20871:     procedure Checkers1Click(Sender: TObject);');
```

```
20872:     procedure TaskMgr1Click(Sender: TObject);');
20873:     procedure WebCam1Click(Sender: TObject);');
20874:     procedure Tutorial31Closure1Click(Sender: TObject);');
20875:     procedure GEOMapView1Click(Sender: TObject);');
20876:
20877:
20878: //-----------------------------------------------------------------------
20879: //*************mX4 Editor SynEdit Tools API ********************************
20880: //-----------------------------------------------------------------------
20881: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
20882: begin
20883:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
20884:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
20885:     Constructor Create( AOwner : TComponent)
20886:     SelStart', 'Integer', iptrw);
20887:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
20888:     Procedure UpdateCaret
20889:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
20890:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
20891:     Procedure BeginUndoBlock
20892:     Procedure BeginUpdate
20893:     Function CaretInView : Boolean
20894:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
20895:     Procedure Clear
20896:     Procedure ClearAll
20897:     Procedure ClearBookMark( BookMark : Integer)
20898:     Procedure ClearSelection
20899:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
20900:     Procedure ClearUndo
20901:     Procedure CopyToClipboard
20902:     Procedure CutToClipboard
20903:     Procedure DoCopyToClipboard( const SText : string)
20904:     Procedure EndUndoBlock
20905:     Procedure EndUpdate
20906:     Procedure EnsureCursorPosVisible
20907:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
20908:     Procedure FindMatchingBracket
20909:     Function GetMatchingBracket : TBufferCoord
20910:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
20911:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
20912:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
20913:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
20914:               : TSynHighlighterAttributes) : boolean
20915:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
20916:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
20917:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
20918:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
20919:     Procedure GotoBookMark( BookMark : Integer)
20920:     Procedure GotoLineAndCenter( ALine : Integer)
20921:     Function IdentChars : TSynIdentChars
20922:     Procedure InvalidateGutter
20923:     Procedure InvalidateGutterLine( aLine : integer)
20924:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
20925:     Procedure InvalidateLine( Line : integer)
20926:     Procedure InvalidateLines( FirstLine, LastLine : integer)
20927:     Procedure InvalidateSelection
20928:     Function IsBookmark( BookMark : integer) : boolean
20929:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
20930:     Procedure LockUndo
20931:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
20932:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
20933:     Function LineToRow( aLine : integer) : integer
20934:     Function RowToLine( aRow : integer) : integer
20935:     Function NextWordPos : TBufferCoord
20936:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
20937:     Procedure PasteFromClipboard
20938:     Function WordStart : TBufferCoord
20939:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
20940:     Function WordEnd : TBufferCoord
20941:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
20942:     Function PrevWordPos : TBufferCoord
20943:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
20944:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
20945:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
20946:     Procedure Redo
20947:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
20948:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
20949:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
20950:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
20951:     Procedure SelectAll
20952:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
20953:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
20954:     Procedure SetDefaultKeystrokes
20955:     Procedure SetSelWord
20956:     Procedure SetWordBlock( Value : TBufferCoord)
20957:     Procedure Undo
20958:     Procedure UnlockUndo
20959:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
20960:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
```

```
20961:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
20962:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
20963:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
20964:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
20965:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
20966:     Procedure AddFocusControl( aControl : TWinControl)
20967:     Procedure RemoveFocusControl( aControl : TWinControl)
20968:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
20969:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
20970:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
20971:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
20972:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
20973:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
20974:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
20975:     Procedure RemoveLinesPointer
20976:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
20977:     Procedure UnHookTextBuffer
20978:     BlockBegin', 'TBufferCoord', iptrw);
20979:     BlockEnd', 'TBufferCoord', iptrw);
20980:     CanPaste', 'Boolean', iptr);
20981:     CanRedo', 'boolean', iptr);
20982:     CanUndo', 'boolean', iptr);
20983:     CaretX', 'Integer', iptrw);
20984:     CaretY', 'Integer', iptrw);
20985:     CaretXY', 'TBufferCoord', iptrw);
20986:     ActiveLineColor', 'TColor', iptrw);
20987:     DisplayX', 'Integer', iptr);
20988:     DisplayY', 'Integer', iptr);
20989:     DisplayXY', 'TDisplayCoord', iptr);
20990:     DisplayLineCount', 'integer', iptr);
20991:     CharsInWindow', 'Integer', iptr);
20992:     CharWidth', 'integer', iptr);
20993:     Font', 'TFont', iptrw);
20994:     GutterWidth', 'Integer', iptr);
20995:     Highlighter', 'TSynCustomHighlighter', iptrw);
20996:     LeftChar', 'Integer', iptrw);
20997:     LineHeight', 'integer', iptr);
20998:     LinesInWindow', 'Integer', iptr);
20999:     LineText', 'string', iptrw);
21000:     Lines', 'TStrings', iptrw);
21001:     Marks', 'TSynEditMarkList', iptr);
21002:     MaxScrollWidth', 'integer', iptrw);
21003:     Modified', 'Boolean', iptrw);
21004:     PaintLock', 'Integer', iptr);
21005:     ReadOnly', 'Boolean', iptrw);
21006:     SearchEngine', 'TSynEditSearchCustom', iptrw);
21007:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
21008:     SelTabBlock', 'Boolean', iptr);
21009:     SelTabLine', 'Boolean', iptr);
21010:     SelText', 'string', iptrw);
21011:     StateFlags', 'TSynStateFlags', iptr);
21012:     Text', 'string', iptrw);
21013:     TopLine', 'Integer', iptrw);
21014:     WordAtCursor', 'string', iptr);
21015:     WordAtMouse', 'string', iptr);
21016:     UndoList', 'TSynEditUndoList', iptr);
21017:     RedoList', 'TSynEditUndoList', iptr);
21018:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
21019:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
21020:     BorderStyle', 'TSynBorderStyle', iptrw);
21021:     ExtraLineSpacing', 'integer', iptrw);
21022:     Gutter', 'TSynGutter', iptrw);
21023:     HideSelection', 'boolean', iptrw);
21024:     InsertCaret', 'TSynEditCaretType', iptrw);
21025:     InsertMode', 'boolean', iptrw);
21026:     IsScrolling', 'Boolean', iptr);
21027:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
21028:     MaxUndo', 'Integer', iptrw);
21029:     Options', 'TSynEditorOptions', iptrw);
21030:     OverwriteCaret', 'TSynEditCaretType', iptrw);
21031:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
21032:     ScrollHintColor', 'TColor', iptrw);
21033:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
21034:     ScrollBars', 'TScrollStyle', iptrw);
21035:     SelectedColor', 'TSynSelectedColor', iptrw);
21036:     SelectionMode', 'TSynSelectionMode', iptrw);
21037:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
21038:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
21039:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
21040:     WordWrapGlyph', 'TSynGlyph', iptrw);
21041:     OnChange', 'TNotifyEvent', iptrw);
21042:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
21043:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
21044:     OnContextHelp', 'TContextHelpEvent', iptrw);
21045:     OnDropFiles', 'TDropFilesEvent', iptrw);
21046:     OnGutterClick', 'TGutterClickEvent', iptrw);
21047:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
21048:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
21049:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
```

```
21050:     OnPaint', 'TPaintEvent', iptrw);
21051:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
21052:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
21053:     OnReplaceText', 'TReplaceTextEvent', iptrw);
21054:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
21055:     OnStatusChange', 'TStatusChangeEvent', iptrw);
21056:     OnPaintTransient', 'TPaintTransient', iptrw);
21057:     OnScroll', 'TScrollEvent', iptrw);
21058:    end;
21059:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
21060:  Function GetPlaceableHighlighters : TSynHighlighterList
21061:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
21062:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
21063:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
21064:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
21065:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
21066:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
21067:  Function ConvertCodeStringToExtended( AString : String) : String
21068:  Function ConvertExtendedToCodeString( AString : String) : String
21069:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
21070:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
21071:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
21072:
21073:   TSynEditorOption = (
21074:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
21075:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
21076:                                // preceding line
21077:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
21078:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
21079:                                //direction any more
21080:     eoDragDropEditing,         //Allows to select a textblock and drag it in document to another location
21081:     eoDropFiles,               //Allows the editor accept OLE file drops
21082:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
21083:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
21084:     eoGroupUndo,               //When undoing/redoing actions,handle all cont.changes same kind in onecall
21085:                                //instead undoing/redoing each command separately
21086:     eoHalfPageScroll,          //By scrolling with page-up/page-down commands,only scroll half page attime
21087:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
21088:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
21089:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
21090:     eoNoCaret,                 //Makes it so the caret is never visible
21091:     eoNoSelection,             //Disables selecting text
21092:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
21093:     eoScrollByOneLess,         //Forces scrolling to be one less
21094:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
21095:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
21096:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
21097:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
21098:     eoShowSpecialChars,        //Shows the special Characters
21099:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
21100:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
21101:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
21102:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
21103:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
21104:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
21105:
21106:     *******************************Important Editor Short Cuts*****************************);
21107:     Double click to select a word and count words with highlightning.
21108:     Triple click to select a line.
21109:     CTRL+SHIFT+click to extend a selection.
21110:     Drag with the ALT key down to select columns of text !!!
21111:     Drag and drop is supported.
21112:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
21113:     Type CTRL+A to select all.
21114:     Type CTRL+N to set a new line.
21115:     Type CTRL+T to delete a line or token. //Tokenizer
21116:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
21117:     Type CTRL+Shift+T to add ToDo in line and list.
21118:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
21119:     Type CTRL[0..9] to jump or get to bookmarks.
21120:     Type Home to position cursor at beginning of current line and End to position it at end of line.
21121:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
21122:     Page Up and Page Down work as expected.
21123:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
21124:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
21125:
21126: {$ Short Key Positions Ctrl<A-Z>: }
21127: def
21128:    <A> Select All
21129:    <B> Count Words
21130:    <C> Copy
21131:    <D> Internet Start
21132:    <E> Script List
21133:    <F> Find
21134:    <G> Goto
21135:    <H> Mark Line
21136:    <I> Interface List
21137:    <J> Code Completion
21138:    <K> Console
```

```
21139:    <L> Interface List Box
21140:    <M> Font Larger -
21141:    <N> New Line
21142:    <O> Open File
21143:    <P> Font Smaller +
21144:    <Q> Quit
21145:    <R> Replace
21146:    <S> Save!
21147:    <T> Delete Line
21148:    <U> Use Case Editor
21149:    <V> Paste
21150:    <W> URI Links
21151:    <X> Reserved for coding use internal
21152:    <Y> Delete Line
21153:    <Z> Undo
21154:
21155: ref
21156:    F1 Help
21157:    F2 Syntax Check
21158:    F3 Search Next
21159:    F4 New Instance
21160:    F5 Line Mark /Breakpoint
21161:    F6 Goto End
21162:    F7 Debug Step Into
21163:    F8 Debug Step Out
21164:    F9 Compile
21165:    F10 Menu
21166:    F11 Word Count Highlight
21167:    F12 Reserved for coding use internal
21168:
21169:    AddRegisteredVariable( it ,integer');  //for closure!!
21170:    AddRegisteredVariable( sr ,string');  //for closure
21171:    AddRegisteredVariable( bt ,boolean');  //for closure
21172:    AddRegisteredVariable( ft ,double');  //for closure
21173:    AddRegisteredVariable( srlist ,TStringlist');  //for closures
21174:
21175:  def ReservedWords: array[0..82] of string =
21176:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
21177:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
21178:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
21179:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
21180:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
21181:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
21182:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
21183:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
21184:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
21185:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
21186:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
21187:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
21188: //-----------------------------------------------------------------------------
21189: //*************End of mX4 Public  Tools API ********************************
21190: //-----------------------------------------------------------------------------
21191:
21192: Amount of Functions: 13556
21193: Amount of Procedures: 8327
21194: Amount of Constructors: 1338
21195: Totals of Calls: 23221
21196: SHA1: Win 3.9.9.100 B1CFFA2139BEF2D27D17C8212331A1E8C7584BD2
21197:
21198: ************************************************************
21199:  Doc Short Manual with 50 Tips!
21200: ************************************************************
21201: - Install: just save your maxboxdef.ini before and then extract the zip file!
21202: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
21203: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21204: - Menu: With <Crtl><F3> you can search for code on examples
21205: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21206: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
21207: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21208:
21209: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21210: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
21211: - Context Menu: You can printout your scripts as a pdf-file or html-export
21212: - Context: You do have a context menu with the right mouse click
21213:
21214: - Menu: With the UseCase Editor you can convert graphic formats too.
21215: - Menu: On menu Options you find Addons as compiled scripts
21216: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
21217: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21218: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21219: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
21220:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
21221:
21222: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21223: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21224: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21225: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
21226: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
21227: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
```

```
21228:          to delete and Click and mark to drag a bookmark
21229: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21230: - IDE: A file info with system and script information you find in menu Program/Information
21231: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21232: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21233: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21234: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21235: - Editor: Set Bookmarks to check your work in app or code
21236: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21237: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
21238: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21239:
21240: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
21241: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
21242: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
21243: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21244: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21245: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
21246: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21247: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21248: - IDE menu /Help/Tools/ open the Task Manager
21249:
21250: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21251: - Add on when no browser is available start /Options/Add ons/Easy Browser
21252: - Add on SOAP Tester with SOP POST File
21253: - Add on IP Protocol Sniffer with List View
21254: - Add on OpenGL mX Robot Demo for android
21255: - Add on Checkers Game, Add on Oscilloscope
21256:
21257: - Menu: Help/Tools as a Tool Section with DOS Opener
21258: - Menu Editor: export the code as RTF File
21259: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21260: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21261: - Context: Auto Detect of Syntax depending on file extension
21262: - Code: some Windows API function start with w in the name like wGetAtomName();
21263: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21264: - IDE File Check with menu ..View/File Changes/...
21265: - Context: Create a Header with Create Header in Navigator List at right window
21266: - Code: use  SysErrorMessage to get a real Error Description, Ex.
21267:         RemoveDir('c:\NoSuchFolder') ;
21268:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
21269: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21270:
21271: - using DLL example in maXbox: //function: {*****************************************}
21272:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21273:                                  cb: DWORD): BOOL; //stdcall;;
21274:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
21275:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21276:      External  'OpenProcess@kernel32.dll stdcall';
21277:
21278: GCC Compile Ex Script
21279: procedure TFormMain_btnCompileClick(Sender: TObject);
21280: begin
21281:   AProcess:= TProcess.Create(Nil);
21282:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
21283:      + ' -o "' + OpenDialog2.FileName + '"';
21284:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21285:     AProcess.Execute;
21286:     Memo2.Lines.BeginUpdate;
21287:     Memo2.Lines.Clear;
21288:     Memo2.Lines.LoadFromStream(AProcess.Output);
21289:     Memo2.Lines.EndUpdate;
21290:   finally
21291:     AProcess.Free;
21292:   end;
21293: end;
21294:
21295: Stopwatch pattern
21296: Time1:= Time;
21297:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
21298:   if initAndStartBoard then
21299:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
21300:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
21301:   PrintF('%d %s',[Trunc((Time-Time1)*24),
21302:                   FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
21303:
21304:    POST git-receive-pack (chunked)
21305: Pushing to https://github.com/maxkleiner/maXbox3.git
21306: To https://github.com/maxkleiner/maXbox3.git
21307:    f127d21..c6a98da  masterbox2 -> masterbox2
21308: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
21309:
21310: History Shell Hell
21311: PCT Precompile Technology , mX4 ScriptStudio
21312: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
21313: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21314: emax layers: system-package-component-unit-class-function-block
21315: new keywords def ref using maXCalcF
21316: UML: use case act class state seq pac comp dep - lib lab
```

```
21317: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21318: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21319: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21320: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21321: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21322: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21323: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21324: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
21325: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21326: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21327: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21328: Inno Install and Setup Routines
21329: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21330: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21331: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21332: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
21333: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
21334: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21335: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
21336:
21337: Ref:
21338: https://unibe-ch.academia.edu/MaxKleiner
21339: http://www.slideshare.net/maxkleiner1
21340: http://www.scribd.com/max_kleiner
21341: http://www.delphiforfun.org/Programs/Utilities/index.htm
21342: http://www.slideshare.net/maxkleiner1
21343: http://s3.amazonaws.com/PreviewLinks/22959.html
21344: http://www.softwareschule.ch/arduino_training.pdf
21345: http://www.jrsoftware.org/isinfo.php
21346: http://www.be-precision.com/products/precision-builder/express/
21347: http://www.blaisepascal.eu/
21348: http://www.delphibasics.co.uk/
21349: http://www.youtube.com/watch?v=av89HAbqAsI
21350: http://www.angelfire.com/hi5/delphizeus/modal.html
21351: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21352: http://delphi.org/2014/01/every-android-api-for-delphi/
21353: https://en.wikipedia.org/wiki/User:Maxkleiner
21354: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
21355:
21356:
21357: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
21358: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21359: =renderBasicSearchNarrative&q=%s';
21360: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
21361: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
21362:
21363: function OpenMap(const Data: string): boolean;
21364: var encURL: string;
21365: begin
21366:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
21367:   try
21368:    //HttpGet(EncodedURL, mapStream);   //WinInet
21369:    Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
21370:    //OpenDoc(Exepath+'openmapx.html');
21371:    S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
21372:   finally
21373:     encURL:= '';
21374:   end;
21375: end;
21376:
21377: procedure GetGEOMap(C_form,apath: string; const Data: string);
21378: var
21379:   encodedURL: string;
21380:   mapStream: TMemoryStream;
21381: begin
21382:   //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPEncode(Data)]);
21383:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
21384:   mapStream:= TMemoryStream.create;
21385:   try
21386:     Wininet_HttpGet(EncodedURL, mapStream);   //WinInet
21387:     mapStream.Position:= 0;
21388:     mapStream.Savetofile(apath);
21389:     // OpenDoc(apath);
21390:     S_ShellExecute(apath,'',seCmdOpen);
21391:   finally
21392:     mapStream.Free;
21393:   end;
21394: end;
21395:
21396:
21397:
21398: ************************************************************
21399:  unit List asm internal end
21400: ************************************************************
21401: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
21402: 02 unit SIRegister_IdStrings                   //Indy Sockets
21403: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
21404: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
21405: 05 unit IFSI_WinForm1puzzle;                   //maXbox
```

```
21406: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
21407: 07 unit RegisterDateTimeLibrary_R(exec);       //Delphi
21408: 08 unit RIRegister_MathMax_Routines(exec);     //Jedi & Delphi
21409: 09 unit RIRegister_IdGlobal_Routines(exec);    //Indy Sockets
21410: 10 unit RIRegister_SysUtils_Routines(Exec);    //Delphi
21411: 11 unit uPSI_IdTCPConnection;                  //Indy some functions
21412: 12 unit uPSCompiler.pas;                       //PS kernel functions
21413: 13 unit uPSI_DBCommon;                         //DB Common_Routines and Types
21414: 14 unit uPSI_Printers.pas                      //Delphi VCL
21415: 15 unit uPSI_MPlayer.pas                      //Delphi VCL
21416: 16 unit uPSC_comobj;                          //COM Functions
21417: 17 unit uPSI_Clipbrd;                         //Delphi VCL
21418: 18 unit Filectrl in IFSI_SysUtils_max;        //VCL Runtime
21419: 19 unit uPSI_SqlExpr;                        //DBX3
21420: 20 unit uPSI_ADODB;                          //ADODB
21421: 21 unit uPSI_StrHlpr;                        //String Helper Routines
21422: 22 unit uPSI_DateUtils;                      //Expansion to DateTimeLib
21423: 23 unit uPSI_FileUtils;                      //Expansion to Sys/File Utils
21424: 24 unit JUtils / gsUtils;                    //Jedi / Metabase
21425: 25 unit JvFunctions_max;                     //Jedi Functions
21426: 26 unit HTTPParser;                          //Delphi VCL
21427: 27 unit HTTPUtil;                           //Delphi VCL
21428: 28 unit uPSI_XMLUtil;                       //Delphi VCL
21429: 29 unit uPSI_SOAPHTTPClient;                //Delphi VCL SOAP WebService V3.5
21430: 30 unit uPSI_Contnrs;                       //Delphi RTL Container of Classes
21431: 31 unit uPSI_MaskUtils;                     //RTL Edit and Mask functions
21432: 32 unit uPSI_MyBigInt;                      //big integer class with Math
21433: 33 unit uPSI_ConvUtils;                     //Delphi VCL Conversions engine
21434: 34 unit Types_Variants;                     //Delphi\Win32\rtl\sys
21435: 35 unit uPSI_IdHashSHA1;                    //Indy Crypto Lib
21436: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
21437: 37 unit uPSI_IdASN1Util;                    //Indy ASN1Utility Routines;
21438: 38 unit uPSI_IdLogFile;                     //Indy Logger from LogBase
21439: 39 unit uPSI_IdIcmpClient;                  //Indy Ping ICMP
21440: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
21441: 41 unit uPSI_FileCtrl;                      //Delphi RTL
21442: 42 unit uPSI_Outline;                       //Delphi VCL
21443: 43 unit uPSI_ScktComp;                      //Delphi RTL
21444: 44 unit uPSI_Calendar;                      //Delphi VCL
21445: 45 unit uPSI_VListView                      //VListView;
21446: 46 unit uPSI_DBGrids;                       //Delphi VCL
21447: 47 unit uPSI_DBCtrls;                       //Delphi VCL
21448: 48 unit ide_debugoutput;                    //maXbox
21449: 49 unit uPSI_ComCtrls;                      //Delphi VCL
21450: 50 unit uPSC_stdctrls+;                     //Delphi VCL
21451: 51 unit uPSI_Dialogs;                       //Delphi VCL
21452: 52 unit uPSI_StdConvs;                      //Delphi RTL
21453: 53 unit uPSI_DBClient;                      //Delphi RTL
21454: 54 unit uPSI_DBPlatform;                    //Delphi RTL
21455: 55 unit uPSI_Provider;                      //Delphi RTL
21456: 56 unit uPSI_FMTBcd;                        //Delphi RTL
21457: 57 unit uPSI_DBCGrids;                      //Delphi VCL
21458: 58 unit uPSI_CDSUtil;                       //MIDAS
21459: 59 unit uPSI_VarHlpr;                       //Delphi RTL
21460: 60 unit uPSI_ExtDlgs;                       //Delphi VCL
21461: 61 unit sdpStopwatch;                       //maXbox
21462: 62 unit uPSI_JclStatistics;                 //JCL
21463: 63 unit uPSI_JclLogic;                      //JCL
21464: 64 unit uPSI_JclMiscel;                     //JCL
21465: 65 unit uPSI_JclMath_max;                   //JCL RTL
21466: 66 unit uPSI_uTPLb_StreamUtils;             //LockBox 3
21467: 67 unit uPSI_MathUtils;                     //BCB
21468: 68 unit uPSI_JclMultimedia;                 //JCL
21469: 69 unit uPSI_WideStrUtils;                  //Delphi API/RTL
21470: 70 unit uPSI_GraphUtil;                     //Delphi RTL
21471: 71 unit uPSI_TypeTrans;                     //Delphi RTL
21472: 72 unit uPSI_HTTPApp;                       //Delphi VCL
21473: 73 unit uPSI_DBWeb;                         //Delphi VCL
21474: 74 unit uPSI_DBBdeWeb;                      //Delphi VCL
21475: 75 unit uPSI_DBXpressWeb;                   //Delphi VCL
21476: 76 unit uPSI_ShadowWnd;                     //Delphi VCL
21477: 77 unit uPSI_ToolWin;                       //Delphi VCL
21478: 78 unit uPSI_Tabs;                          //Delphi VCL
21479: 79 unit uPSI_JclGraphUtils;                 //JCL
21480: 80 unit uPSI_JclCounter;                    //JCL
21481: 81 unit uPSI_JclSysInfo;                    //JCL
21482: 82 unit uPSI_JclSecurity;                   //JCL
21483: 83 unit uPSI_JclFileUtils;                  //JCL
21484: 84 unit uPSI_IdUserAccounts;                //Indy
21485: 85 unit uPSI_IdAuthentication;              //Indy
21486: 86 unit uPSI_uTPLb_AES;                     //LockBox 3
21487: 87 unit uPSI_IdHashSHA1;                    //LockBox 3
21488: 88 unit uTPLb_BlockCipher;                  //LockBox 3
21489: 89 unit uPSI_ValEdit.pas;                   //Delphi VCL
21490: 90 unit uPSI_JvVCLUtils;                    //JCL
21491: 91 unit uPSI_JvDBUtil;                      //JCL
21492: 92 unit uPSI_JvDBUtils;                     //JCL
21493: 93 unit uPSI_JvAppUtils;                    //JCL
21494: 94 unit uPSI_JvCtrlUtils;                   //JCL
```

```
21495: 95 unit uPSI_JvFormToHtml;                    //JCL
21496: 96 unit uPSI_JvParsing;                       //JCL
21497: 97 unit uPSI_SerDlgs;                         //Toolbox
21498: 98 unit uPSI_Serial;                          //Toolbox
21499: 99 unit uPSI_JvComponent;                     //JCL
21500: 100 unit uPSI_JvCalc;                         //JCL
21501: 101 unit uPSI_JvBdeUtils;                     //JCL
21502: 102 unit uPSI_JvDateUtil;                     //JCL
21503: 103 unit uPSI_JvGenetic;                      //JCL
21504: 104 unit uPSI_JclBase;                        //JCL
21505: 105 unit uPSI_JvUtils;                        //JCL
21506: 106 unit uPSI_JvStrUtil;                      //JCL
21507: 107 unit uPSI_JvStrUtils;                     //JCL
21508: 108 unit uPSI_JvFileUtil;                     //JCL
21509: 109 unit uPSI_JvMemoryInfos;                  //JCL
21510: 110 unit uPSI_JvComputerInfo;                 //JCL
21511: 111 unit uPSI_JvgCommClasses;                 //JCL
21512: 112 unit uPSI_JvgLogics;                      //JCL
21513: 113 unit uPSI_JvLED;                          //JCL
21514: 114 unit uPSI_JvTurtle;                       //JCL
21515: 115 unit uPSI_SortThds; unit uPSI_ThSort;     //maXbox
21516: 116 unit uPSI_JvgUtils;                       //JCL
21517: 117 unit uPSI_JvExprParser;                   //JCL
21518: 118 unit uPSI_HexDump;                        //Borland
21519: 119 unit uPSI_DBLogDlg;                       //VCL
21520: 120 unit uPSI_SqlTimSt;                       //RTL
21521: 121 unit uPSI_JvHtmlParser;                   //JCL
21522: 122 unit uPSI_JvgXMLSerializer;               //JCL
21523: 123 unit uPSI_JvJCLUtils;                     //JCL
21524: 124 unit uPSI_JvStrings;                      //JCL
21525: 125 unit uPSI_uTPLb_IntegerUtils;             //TurboPower
21526: 126 unit uPSI_uTPLb_HugeCardinal;             //TurboPower
21527: 127 unit uPSI_uTPLb_HugeCardinalUtils;        //TurboPower
21528: 128 unit uPSI_SynRegExpr;                     //SynEdit
21529: 129 unit uPSI_StUtils;                        //SysTools4
21530: 130 unit uPSI_StToHTML;                       //SysTools4
21531: 131 unit uPSI_StStrms;                        //SysTools4
21532: 132 unit uPSI_StFIN;                          //SysTools4
21533: 133 unit uPSI_StAstroP;                       //SysTools4
21534: 134 unit uPSI_StStat;                         //SysTools4
21535: 135 unit uPSI_StNetCon;                       //SysTools4
21536: 136 unit uPSI_StDecMth;                       //SysTools4
21537: 137 unit uPSI_StOStr;                         //SysTools4
21538: 138 unit uPSI_StPtrns;                        //SysTools4
21539: 139 unit uPSI_StNetMsg;                       //SysTools4
21540: 140 unit uPSI_StMath;                         //SysTools4
21541: 141 unit uPSI_StExpEng;                       //SysTools4
21542: 142 unit uPSI_StCRC;                          //SysTools4
21543: 143 unit uPSI_StExport,                       //SysTools4
21544: 144 unit uPSI_StExpLog,                       //SysTools4
21545: 145 unit uPSI_ActnList;                       //Delphi VCL
21546: 146 unit uPSI_jpeg;                           //Borland
21547: 147 unit uPSI_StRandom;                       //SysTools4
21548: 148 unit uPSI_StDict;                         //SysTools4
21549: 149 unit uPSI_StBCD;                          //SysTools4
21550: 150 unit uPSI_StTxtDat;                       //SysTools4
21551: 151 unit uPSI_StRegEx;                        //SysTools4
21552: 152 unit uPSI_IMouse;                         //VCL
21553: 153 unit uPSI_SyncObjs;                       //VCL
21554: 154 unit uPSI_AsyncCalls;                     //Hausladen
21555: 155 unit uPSI_ParallelJobs;                   //Saraiva
21556: 156 unit uPSI_Variants;                       //VCL
21557: 157 unit uPSI_VarCmplx;                       //VCL Wolfram
21558: 158 unit uPSI_DTDSchema;                      //VCL
21559: 159 unit uPSI_ShLwApi;                        //Brakel
21560: 160 unit uPSI_IBUtils;                        //VCL
21561: 161 unit uPSI_CheckLst;                       //VCL
21562: 162 unit uPSI_JvSimpleXml;                    //JCL
21563: 163 unit uPSI_JclSimpleXml;                   //JCL
21564: 164 unit uPSI_JvXmlDatabase;                  //JCL
21565: 165 unit uPSI_JvMaxPixel;                     //JCL
21566: 166 unit uPSI_JvItemsSearchs;                 //JCL
21567: 167 unit uPSI_StExpEng2;                      //SysTools4
21568: 168 unit uPSI_StGenLog;                       //SysTools4
21569: 169 unit uPSI_JvLogFile;                      //Jcl
21570: 170 unit uPSI_CPort;                          //ComPort Lib v4.11
21571: 171 unit uPSI_CPortCtl;                       //ComPort
21572: 172 unit uPSI_CPortEsc;                       //ComPort
21573: 173 unit BarCodeScaner;                       //ComPort
21574: 174 unit uPSI_JvGraph;                        //JCL
21575: 175 unit uPSI_JvComCtrls;                     //JCL
21576: 176 unit uPSI_GUITesting;                     //D Unit
21577: 177 unit uPSI_JvFindFiles;                    //JCL
21578: 178 unit uPSI_StSystem;                       //SysTools4
21579: 179 unit uPSI_JvKeyboardStates;               //JCL
21580: 180 unit uPSI_JvMail;                         //JCL
21581: 181 unit uPSI_JclConsole;                     //JCL
21582: 182 unit uPSI_JclLANMan;                      //JCL
21583: 183 unit uPSI_IdCustomHTTPServer;             //Indy
```

```
21584: 184 unit IdHTTPServer                          //Indy
21585: 185 unit uPSI_IdTCPServer;                     //Indy
21586: 186 unit uPSI_IdSocketHandle;                  //Indy
21587: 187 unit uPSI_IdIOHandlerSocket;               //Indy
21588: 188 unit IdIOHandler;                          //Indy
21589: 189 unit uPSI_cutils;                          //Bloodshed
21590: 190 unit uPSI_BoldUtils;                       //boldsoft
21591: 191 unit uPSI_IdSimpleServer;                  //Indy
21592: 192 unit uPSI_IdSSLOpenSSL;                    //Indy
21593: 193 unit uPSI_IdMultipartFormData;             //Indy
21594: 194 unit uPSI_SynURIOpener;                    //SynEdit
21595: 195 unit uPSI_PerlRegEx;                       //PCRE
21596: 196 unit uPSI_IdHeaderList;                    //Indy
21597: 197 unit uPSI_StFirst;                         //SysTools4
21598: 198 unit uPSI_JvCtrls;                         //JCL
21599: 199 unit uPSI_IdTrivialFTPBase;                //Indy
21600: 200 unit uPSI_IdTrivialFTP;                    //Indy
21601: 201 unit uPSI_IdUDPBase;                       //Indy
21602: 202 unit uPSI_IdUDPClient;                     //Indy
21603: 203 unit uPSI_utypes;                          //for DMath.DLL
21604: 204 unit uPSI_ShellAPI;                        //Borland
21605: 205 unit uPSI_IdRemoteCMDClient;               //Indy
21606: 206 unit uPSI_IdRemoteCMDServer;               //Indy
21607: 207 unit IdRexecServer;                        //Indy
21608: 208 unit IdRexec; (unit uPSI_IdRexec;)         //Indy
21609: 209 unit IdUDPServer;                          //Indy
21610: 210 unit IdTimeUDPServer;                      //Indy
21611: 211 unit IdTimeServer;                         //Indy
21612: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)   //Indy
21613: 213 unit uPSI_IdIPWatch;                       //Indy
21614: 214 unit uPSI_IdIrcServer;                     //Indy
21615: 215 unit uPSI_IdMessageCollection;             //Indy
21616: 216 unit uPSI_cPEM;                            //Fundamentals 4
21617: 217 unit uPSI_cFundamentUtils;                 //Fundamentals 4
21618: 218 unit uPSI_uwinplot;                        //DMath
21619: 219 unit uPSI_xrtl_util_CPUUtils;              //ExtentedRTL
21620: 220 unit uPSI_GR32_System;                     //Graphics32
21621: 221 unit uPSI_cFileUtils;                      //Fundamentals 4
21622: 222 unit uPSI_cDateTime; (timemachine)         //Fundamentals 4
21623: 223 unit uPSI_cTimers; (high precision timer)  //Fundamentals 4
21624: 224 unit uPSI_cRandom;                         //Fundamentals 4
21625: 225 unit uPSI_ueval;                           //DMath
21626: 226 unit uPSI_xrtl_net_URIUtils;               //ExtendedRTL
21627: 227 unit xrtl_net_URIUtils;                    //ExtendedRTL
21628: 228 unit uPSI_ufft;  (FFT)                     //DMath
21629: 229 unit uPSI_DBXChannel;                      //Delphi
21630: 230 unit uPSI_DBXIndyChannel;                  //Delphi Indy
21631: 231 unit uPSI_xrtl_util_COMCat;                //ExtendedRTL
21632: 232 unit uPSI_xrtl_util_StrUtils;              //ExtendedRTL
21633: 233 unit uPSI_xrtl_util_VariantUtils;          //ExtendedRTL
21634: 234 unit uPSI_xrtl_util_FileUtils;             //ExtendedRTL
21635: 235 unit xrtl_util_Compat;                     //ExtendedRTL
21636: 236 unit uPSI_OleAuto;                         //Borland
21637: 237 unit uPSI_xrtl_util_COMUtils;              //ExtendedRTL
21638: 238 unit uPSI_CmAdmCtl;                        //Borland
21639: 239 unit uPSI_ValEdit2;                        //VCL
21640: 240 unit uPSI_GR32;  //Graphics32             //Graphics32
21641: 241 unit uPSI_GR32_Image;                      //Graphics32
21642: 242 unit uPSI_xrtl_util_TimeUtils;             //ExtendedRTL
21643: 243 unit uPSI_xrtl_util_TimeZone;              //ExtendedRTL
21644: 244 unit uPSI_xrtl_util_TimeStamp;             //ExtendedRTL
21645: 245 unit uPSI_xrtl_util_Map;                   //ExtendedRTL
21646: 246 unit uPSI_xrtl_util_Set;                   //ExtendedRTL
21647: 247 unit uPSI_CPortMonitor;                    //ComPort
21648: 248 unit uPSI_StIniStm;                        //SysTools4
21649: 249 unit uPSI_GR32_ExtImage;                   //Graphics32
21650: 250 unit uPSI_GR32_OrdinalMaps;                //Graphics32
21651: 251 unit uPSI_GR32_Rasterizers;                //Graphics32
21652: 252 unit uPSI_xrtl_util_Exception;             //ExtendedRTL
21653: 253 unit uPSI_xrtl_util_Value;                 //ExtendedRTL
21654: 254 unit uPSI_xrtl_util_Compare;               //ExtendedRTL
21655: 255 unit uPSI_FlatSB;                          //VCL
21656: 256 unit uPSI_JvAnalogClock;                   //JCL
21657: 257 unit uPSI_JvAlarms;                        //JCL
21658: 258 unit uPSI_JvSQLS;                          //JCL
21659: 259 unit uPSI_JvDBSecur;                       //JCL
21660: 260 unit uPSI_JvDBQBE;                         //JCL
21661: 261 unit uPSI_JvStarfield;                     //JCL
21662: 262 unit uPSI_JVCLMiscal;                      //JCL
21663: 263 unit uPSI_JvProfiler32;                    //JCL
21664: 264 unit uPSI_JvDirectories,                   //JCL
21665: 265 unit uPSI_JclSchedule,                     //JCL
21666: 266 unit uPSI_JclSvcCtrl,                      //JCL
21667: 267 unit uPSI_JvSoundControl,                  //JCL
21668: 268 unit uPSI_JvBDESQLScript,                  //JCL
21669: 269 unit uPSI_JvgDigits,                       //JCL>
21670: 270 unit uPSI_ImgList;                         //TCustomImageList
21671: 271 unit uPSI_JclMIDI;                         //JCL>
21672: 272 unit uPSI_JclWinMidi;                      //JCL>
```

```
21673: 273 unit uPSI_JclNTFS;                        //JCL>
21674: 274 unit uPSI_JclAppInst;                     //JCL>
21675: 275 unit uPSI_JvRle;                          //JCL>
21676: 276 unit uPSI_JvRas32;                        //JCL>
21677: 277 unit uPSI_JvImageDrawThread,              //JCL>
21678: 278 unit uPSI_JvImageWindow,                  //JCL>
21679: 279 unit uPSI_JvTransparentForm;              //JCL>
21680: 280 unit uPSI_JvWinDialogs;                   //JCL>
21681: 281 unit uPSI_JvSimLogic,                    //JCL>
21682: 282 unit uPSI_JvSimIndicator,                //JCL>
21683: 283 unit uPSI_JvSimPID,                      //JCL>
21684: 284 unit uPSI_JvSimPIDLinker,                //JCL>
21685: 285 unit uPSI_IdRFCReply;                    //Indy
21686: 286 unit uPSI_IdIdent;                       //Indy
21687: 287 unit uPSI_IdIdentServer;                 //Indy
21688: 288 unit uPSI_JvPatchFile;                   //JCL
21689: 289 unit uPSI_StNetPfm;                      //SysTools4
21690: 290 unit uPSI_StNet;                         //SysTools4
21691: 291 unit uPSI_JclPeImage;                    //JCL
21692: 292 unit uPSI_JclPrint;                      //JCL
21693: 293 unit uPSI_JclMime;                       //JCL
21694: 294 unit uPSI_JvRichEdit;                    //JCL
21695: 295 unit uPSI_JvDBRichEd;                    //JCL
21696: 296 unit uPSI_JvDice;                        //JCL
21697: 297 unit uPSI_JvFloatEdit;                   //JCL 3.9.8
21698: 298 unit uPSI_JvDirFrm;                      //JCL
21699: 299 unit uPSI_JvDualList;                    //JCL
21700: 300 unit uPSI_JvSwitch;                      ////JCL
21701: 301 unit uPSI_JvTimerLst;                    ////JCL
21702: 302 unit uPSI_JvMemTable;                    //JCL
21703: 303 unit uPSI_JvObjStr;                      //JCL
21704: 304 unit uPSI_StLArr;                        //SysTools4
21705: 305 unit uPSI_StWmDCpy;                      //SysTools4
21706: 306 unit uPSI_StText;                        //SysTools4
21707: 307 unit uPSI_StNTLog;                       //SysTools4
21708: 308 unit uPSI_xrtl_math_Integer;             //ExtendedRTL
21709: 309 unit uPSI_JvImagPrvw;                    //JCL
21710: 310 unit uPSI_JvFormPatch;                   //JCL
21711: 311 unit uPSI_JvPicClip;                     //JCL
21712: 312 unit uPSI_JvDataConv;                    //JCL
21713: 313 unit uPSI_JvCpuUsage;                    //JCL
21714: 314 unit uPSI_JclUnitConv_mX2;               //JCL
21715: 315 unit JvDualListForm;                     //JCL
21716: 316 unit uPSI_JvCpuUsage2;                   //JCL
21717: 317 unit uPSI_JvParserForm;                  //JCL
21718: 318 unit uPSI_JvJanTreeView;                 //JCL
21719: 319 unit uPSI_JvTransLED;                    //JCL
21720: 320 unit uPSI_JvPlaylist;                    //JCL
21721: 321 unit uPSI_JvFormAutoSize;                //JCL
21722: 322 unit uPSI_JvYearGridEditForm;            //JCL
21723: 323 unit uPSI_JvMarkupCommon;                //JCL
21724: 324 unit uPSI_JvChart;                       //JCL
21725: 325 unit uPSI_JvXPCore;                      //JCL
21726: 326 unit uPSI_JvXPCoreUtils;                 //JCL
21727: 327 unit uPSI_StatsClasses;                  //mX4
21728: 328 unit uPSI_ExtCtrls2;                     //VCL
21729: 329 unit uPSI_JvUrlGrabbers;                 //JCL
21730: 330 unit uPSI_JvXmlTree;                     //JCL
21731: 331 unit uPSI_JvWavePlayer;                  //JCL
21732: 332 unit uPSI_JvUnicodeCanvas;               //JCL
21733: 333 unit uPSI_JvTFUtils;                     //JCL
21734: 334 unit uPSI_IdServerIOHandler;             //Indy
21735: 335 unit uPSI_IdServerIOHandlerSocket;       //Indy
21736: 336 unit uPSI_IdMessageCoder;                //Indy
21737: 337 unit uPSI_IdMessageCoderMIME;            //Indy
21738: 338 unit uPSI_IdMIMETypes;                   //Indy
21739: 339 unit uPSI_JvConverter;                   //JCL
21740: 340 unit uPSI_JvCsvParse;                    //JCL
21741: 341 unit uPSI_umath;  unit uPSI_ugamma;      //DMath
21742: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
21743: 343 unit uPSI_JvDBGridExport;                //JCL
21744: 344 unit uPSI_JvgExport;                     //JCL
21745: 345 unit uPSI_JvSerialMaker;                 //JCL
21746: 346 unit uPSI_JvWin32;                       //JCL
21747: 347 unit uPSI_JvPaintFX;                     //JCL
21748: 348 unit uPSI_JvOracleDataSet; (beta)        //JCL
21749: 349 unit uPSI_JvValidators; (preview)        //JCL
21750: 350 unit uPSI_JvNTEventLog;                  //JCL
21751: 351 unit uPSI_ShellZipTool;                  //mX4
21752: 352 unit uPSI_JvJoystick;                    //JCL
21753: 353 unit uPSI_JvMailSlots;                   //JCL
21754: 354 unit uPSI_JclComplex;                    //JCL
21755: 355 unit uPSI_SynPdf;                        //Synopse
21756: 356 unit uPSI_Registry;                      //VCL
21757: 357 unit uPSI_TlHelp32;                      //VCL
21758: 358 unit uPSI_JclRegistry;                   //JCL
21759: 359 unit uPSI_JvAirBrush;                    //JCL
21760: 360 unit uPSI_mORMotReport;                  //Synopse
21761: 361 unit uPSI_JclLocales;                    //JCL
```

```
21762: 362 unit uPSI_SynEdit;                    //SynEdit
21763: 363 unit uPSI_SynEditTypes;               //SynEdit
21764: 364 unit uPSI_SynMacroRecorder;           //SynEdit
21765: 365 unit uPSI_LongIntList;                //SynEdit
21766: 366 unit uPSI_devcutils;                  //DevC
21767: 367 unit uPSI_SynEditMiscClasses;         //SynEdit
21768: 368 unit uPSI_SynEditRegexSearch;         //SynEdit
21769: 369 unit uPSI_SynEditHighlighter;         //SynEdit
21770: 370 unit uPSI_SynHighlighterPas;          //SynEdit
21771: 371 unit uPSI_JvSearchFiles;              //JCL
21772: 372 unit uPSI_SynHighlighterAny;          //Lazarus
21773: 373 unit uPSI_SynEditKeyCmds;             //SynEdit
21774: 374 unit uPSI_SynEditMiscProcs,           //SynEdit
21775: 375 unit uPSI_SynEditKbdHandler           //SynEdit
21776: 376 unit uPSI_JvAppInst,                  //JCL
21777: 377 unit uPSI_JvAppEvent;                 //JCL
21778: 378 unit uPSI_JvAppCommand;               //JCL
21779: 379 unit uPSI_JvAnimTitle;                //JCL
21780: 380 unit uPSI_JvAnimatedImage;            //JCL
21781: 381 unit uPSI_SynEditExport;              //SynEdit
21782: 382 unit uPSI_SynExportHTML;              //SynEdit
21783: 383 unit uPSI_SynExportRTF;               //SynEdit
21784: 384 unit uPSI_SynEditSearch;              //SynEdit
21785: 385 unit uPSI_fMain_back                  //maXbox;
21786: 386 unit uPSI_JvZoom;                     //JCL
21787: 387 unit uPSI_PMrand;                     //PM
21788: 388 unit uPSI_JvSticker;                  //JCL
21789: 389 unit uPSI_XmlVerySimple;              //mX4
21790: 390 unit uPSI_Services;                   //ExtPascal
21791: 391 unit uPSI_ExtPascalUtils;             //ExtPascal
21792: 392 unit uPSI_SocketsDelphi;              //ExtPascal
21793: 393 unit uPSI_StBarC;                     //SysTools
21794: 394 unit uPSI_StDbBarC;                   //SysTools
21795: 395 unit uPSI_StBarPN;                    //SysTools
21796: 396 unit uPSI_StDbPNBC;                   //SysTools
21797: 397 unit uPSI_StDb2DBC;                   //SysTools
21798: 398 unit uPSI_StMoney;                    //SysTools
21799: 399 unit uPSI_JvForth;                    //JCL
21800: 400 unit uPSI_RestRequest;                //mX4
21801: 401 unit uPSI_HttpRESTConnectionIndy;     //mX4
21802: 402 unit uPSI_JvXmlDatabase; //update     //JCL
21803: 403 unit uPSI_StAstro;                    //SysTools
21804: 404 unit uPSI_StSort;                     //SysTools
21805: 405 unit uPSI_StDate;                     //SysTools
21806: 406 unit uPSI_StDateSt;                   //SysTools
21807: 407 unit uPSI_StBase;                     //SysTools
21808: 408 unit uPSI_StVInfo;                    //SysTools
21809: 409 unit uPSI_JvBrowseFolder;             //JCL
21810: 410 unit uPSI_JvBoxProcs;                 //JCL
21811: 411 unit uPSI_urandom; (unit uranuvag;)   //DMath
21812: 412 unit uPSI_usimann; (unit ugenalg;)    //DMath
21813: 413 unit uPSI_JvHighlighter;              //JCL
21814: 414 unit uPSI_Diff;                       //mX4
21815: 415 unit uPSI_SpringWinAPI;               //DSpring
21816: 416 unit uPSI_StBits;                     //SysTools
21817: 417 unit uPSI_TomDBQue;                   //mX4
21818: 418 unit uPSI_MultilangTranslator;        //mX4
21819: 419 unit uPSI_HyperLabel;                 //mX4
21820: 420 unit uPSI_Starter;                    //mX4
21821: 421 unit uPSI_FileAssocs;                 //devC
21822: 422 unit uPSI_devFileMonitorX;            //devC
21823: 423 unit uPSI_devrun;                     //devC
21824: 424 unit uPSI_devExec;                    //devC
21825: 425 unit uPSI_oysUtils;                   //devC
21826: 426 unit uPSI_DosCommand;                 //devC
21827: 427 unit uPSI_CppTokenizer;               //devC
21828: 428 unit uPSI_JvHLParser;                 //devC
21829: 429 unit uPSI_JclMapi;                    //JCL
21830: 430 unit uPSI_JclShell;                   //JCL
21831: 431 unit uPSI_JclCOM;                     //JCL
21832: 432 unit uPSI_GR32_Math;                  //Graphics32
21833: 433 unit uPSI_GR32_LowLevel;              //Graphics32
21834: 434 unit uPSI_SimpleHl;                   //mX4
21835: 435 unit uPSI_GR32_Filters,               //Graphics32
21836: 436 unit uPSI_GR32_VectorMaps;            //Graphics32
21837: 437 unit uPSI_cXMLFunctions;              //Fundamentals 4
21838: 438 unit uPSI_JvTimer;                    //JCL
21839: 439 unit uPSI_cHTTPUtils;                 //Fundamentals 4
21840: 440 unit uPSI_cTLSUtils;                  //Fundamentals 4
21841: 441 unit uPSI_JclGraphics;                //JCL
21842: 442 unit uPSI_JclSynch;                   //JCL
21843: 443 unit uPSI_IdTelnet;                   //Indy
21844: 444 unit uPSI_IdTelnetServer,             //Indy
21845: 445 unit uPSI_IdEcho,                     //Indy
21846: 446 unit uPSI_IdEchoServer,               //Indy
21847: 447 unit uPSI_IdEchoUDP,                  //Indy
21848: 448 unit uPSI_IdEchoUDPServer,            //Indy
21849: 449 unit uPSI_IdSocks,                    //Indy
21850: 450 unit uPSI_IdAntiFreezeBase;           //Indy
```

```
21851: 451 unit uPSI_IdHostnameServer;                //Indy
21852: 452 unit uPSI_IdTunnelCommon,                  //Indy
21853: 453 unit uPSI_IdTunnelMaster,                  //Indy
21854: 454 unit uPSI_IdTunnelSlave,                   //Indy
21855: 455 unit uPSI_IdRSH,                           //Indy
21856: 456 unit uPSI_IdRSHServer,                     //Indy
21857: 457 unit uPSI_Spring_Cryptography_Utils;       //Spring4Delphi
21858: 458 unit uPSI_MapReader,                      //devC
21859: 459 unit uPSI_LibTar,                         //devC
21860: 460 unit uPSI_IdStack;                        //Indy
21861: 461 unit uPSI_IdBlockCipherIntercept;         //Indy
21862: 462 unit uPSI_IdChargenServer;                //Indy
21863: 463 unit uPSI_IdFTPServer,                    //Indy
21864: 464 unit uPSI_IdException,                    //Indy
21865: 465 unit uPSI_utexplot;                       //DMath
21866: 466 unit uPSI_uwinstr;                        //DMath
21867: 467 unit uPSI_VarRecUtils;                    //devC
21868: 468 unit uPSI_JvStringListToHtml,             //JCL
21869: 469 unit uPSI_JvStringHolder,                 //JCL
21870: 470 unit uPSI_IdCoder;                        //Indy
21871: 471 unit uPSI_SynHighlighterDfm;              //Synedit
21872: 472 unit uHighlighterProcs; in 471            //Synedit
21873: 473 unit uPSI_LazFileUtils,                   //LCL
21874: 474 unit uPSI_IDECmdLine;                     //LCL
21875: 475 unit uPSI_lazMasks;                       //LCL
21876: 476 unit uPSI_ip_misc;                        //mX4
21877: 477 unit uPSI_Barcode;                        //LCL
21878: 478 unit uPSI_SimpleXML;                      //LCL
21879: 479 unit uPSI_JclIniFiles;                    //JCL
21880: 480 unit uPSI_D2XXUnit;  {$X-}                //FTDI
21881: 481 unit uPSI_JclDateTime;                    //JCL
21882: 482 unit uPSI_JclEDI;                         //JCL
21883: 483 unit uPSI_JclMiscel2;                     //JCL
21884: 484 unit uPSI_JclValidation;                  //JCL
21885: 485 unit uPSI_JclAnsiStrings; {-PString}      //JCL
21886: 486 unit uPSI_SynEditMiscProcs2;              //Synedit
21887: 487 unit uPSI_JclStreams;                     //JCL
21888: 488 unit uPSI_QRCode;                         //mX4
21889: 489 unit uPSI_BlockSocket;                    //ExtPascal
21890: 490 unit uPSI_Masks,Utils                     //VCL
21891: 491 unit uPSI_synautil;                       //Synapse!
21892: 492 unit uPSI_JclMath_Class;                  //JCL RTL
21893: 493 unit ugamdist; //Gamma function          //DMath
21894: 494 unit uibeta, ucorrel; //IBeta            //DMath
21895: 495 unit uPSI_SRMgr;                          //mX4
21896: 496 unit uPSI_HotLog;                         //mX4
21897: 497 unit uPSI_DebugBox;                       //mX4
21898: 498 unit uPSI_ustrings;                       //DMath
21899: 499 unit uPSI_uregtest;                       //DMath
21900: 500 unit uPSI_usimplex;                       //DMath
21901: 501 unit uPSI_uhyper;                         //DMath
21902: 502 unit uPSI_IdHL7;                          //Indy
21903: 503 unit uPSI_IdIPMCastBase,                  //Indy
21904: 504 unit uPSI_IdIPMCastServer;                //Indy
21905: 505 unit uPSI_IdIPMCastClient;                //Indy
21906: 506 unit uPSI_unlfit; //nlregression         //DMath
21907: 507 unit uPSI_IdRawHeaders;                   //Indy
21908: 508 unit uPSI_IdRawClient;                    //Indy
21909: 509 unit uPSI_IdRawFunctions;                 //Indy
21910: 510 unit uPSI_IdTCPStream;                    //Indy
21911: 511 unit uPSI_IdSNPP;                         //Indy
21912: 512 unit uPSI_St2DBarC;                       //SysTools
21913: 513 unit uPSI_ImageWin;  //FTL               //VCL
21914: 514 unit uPSI_CustomDrawTreeView; //FTL      //VCL
21915: 515 unit uPSI_GraphWin;  //FTL               //VCL
21916: 516 unit uPSI_actionMain;  //FTL             //VCL
21917: 517 unit uPSI_StSpawn;                        //SysTools
21918: 518 unit uPSI_CtlPanel;                       //VCL
21919: 519 unit uPSI_IdLPR;                          //Indy
21920: 520 unit uPSI_SockRequestInterpreter;         //Indy
21921: 521 unit uPSI_ulambert;                       //DMath
21922: 522 unit uPSI_ucholesk;                       //DMath
21923: 523 unit uPSI_SimpleDS;                       //VCL
21924: 524 unit uPSI_DBXSqlScanner;                  //VCL
21925: 525 unit uPSI_DBXMetaDataUtil;                //VCL
21926: 526 unit uPSI_Chart;                          //TEE
21927: 527 unit uPSI_TeeProcs;                       //TEE
21928: 528 unit mXBDEUtils;                          //mX4
21929: 529 unit uPSI_MDIEdit;                        //VCL
21930: 530 unit uPSI_CopyPrsr;                       //VCL
21931: 531 unit uPSI_SockApp;                        //VCL
21932: 532 unit uPSI_AppEvnts;                       //VCL
21933: 533 unit uPSI_ExtActns;                       //VCL
21934: 534 unit uPSI_TeEngine;                       //TEE
21935: 535 unit uPSI_CoolMain; //browser             //VCL
21936: 536 unit uPSI_StCRC;                          //SysTools
21937: 537 unit uPSI_StDecMth2;                      //SysTools
21938: 538 unit uPSI_frmExportMain;                  //Synedit
21939: 539 unit uPSI_SynDBEdit;                      //Synedit
```

```
21940: 540 unit uPSI_SynEditWildcardSearch;          //Synedit
21941: 541 unit uPSI_BoldComUtils;                   //BOLD
21942: 542 unit uPSI_BoldIsoDateTime;                //BOLD
21943: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils    //BOLD
21944: 544 unit uPSI_BoldXMLRequests;               //BOLD
21945: 545 unit uPSI_BoldStringList;                //BOLD
21946: 546 unit uPSI_BoldFileHandler;               //BOLD
21947: 547 unit uPSI_BoldContainers;                //BOLD
21948: 548 unit uPSI_BoldQueryUserDlg;              //BOLD
21949: 549 unit uPSI_BoldWinINet;                   //BOLD
21950: 550 unit uPSI_BoldQueue;                     //BOLD
21951: 551 unit uPSI_JvPcx;                         //JCL
21952: 552 unit uPSI_IdWhois;                       //Indy
21953: 553 unit uPSI_IdWhoIsServer;                 //Indy
21954: 554 unit uPSI_IdGopher;                      //Indy
21955: 555 unit uPSI_IdDateTimeStamp;               //Indy
21956: 556 unit uPSI_IdDayTimeServer;               //Indy
21957: 557 unit uPSI_IdDayTimeUDP;                  //Indy
21958: 558 unit uPSI_IdDayTimeUDPServer;            //Indy
21959: 559 unit uPSI_IdDICTServer;                  //Indy
21960: 560 unit uPSI_IdDiscardServer;               //Indy
21961: 561 unit uPSI_IdDiscardUDPServer;            //Indy
21962: 562 unit uPSI_IdMappedFTP;                   //Indy
21963: 563 unit uPSI_IdMappedPortTCP;               //Indy
21964: 564 unit uPSI_IdGopherServer;                //Indy
21965: 565 unit uPSI_IdQotdServer;                  //Indy
21966: 566 unit uPSI_JvRgbToHtml;                   //JCL
21967: 567 unit uPSI_JvRemLog,                      //JCL
21968: 568 unit uPSI_JvSysComp;                     //JCL
21969: 569 unit uPSI_JvTMTL;                        //JCL
21970: 570 unit uPSI_JvWinampAPI;                   //JCL
21971: 571 unit uPSI_MSysUtils;                     //mX4
21972: 572 unit uPSI_ESBMaths;                      //ESB
21973: 573 unit uPSI_ESBMaths2;                     //ESB
21974: 574 unit uPSI_uLkJSON;                       //Lk
21975: 575 unit uPSI_ZURL;  //Zeos                  //Zeos
21976: 576 unit uPSI_ZSysUtils;                     //Zeos
21977: 577 unit unaUtils internals                  //UNA
21978: 578 unit uPSI_ZMatchPattern;                 //Zeos
21979: 579 unit uPSI_ZClasses;                      //Zeos
21980: 580 unit uPSI_ZCollections;                  //Zeos
21981: 581 unit uPSI_ZEncoding;                     //Zeos
21982: 582 unit uPSI_IdRawBase;                     //Indy
21983: 583 unit uPSI_IdNTLM;                        //Indy
21984: 584 unit uPSI_IdNNTP;                        //Indy
21985: 585 unit uPSI_usniffer; //PortScanForm       //mX4
21986: 586 unit uPSI_IdCoderMIME;                   //Indy
21987: 587 unit uPSI_IdCoderUUE;                    //Indy
21988: 588 unit uPSI_IdCoderXXE;                    //Indy
21989: 589 unit uPSI_IdCoder3to4;                   //Indy
21990: 590 unit uPSI_IdCookie;                      //Indy
21991: 591 unit uPSI_IdCookieManager;               //Indy
21992: 592 unit uPSI_WDosSocketUtils;               //WDos
21993: 593 unit uPSI_WDosPlcUtils;                  //WDos
21994: 594 unit uPSI_WDosPorts;                     //WDos
21995: 595 unit uPSI_WDosResolvers;                 //WDos
21996: 596 unit uPSI_WDosTimers;                    //WDos
21997: 597 unit uPSI_WDosPlcs;                      //WDos
21998: 598 unit uPSI_WDosPneumatics;                //WDos
21999: 599 unit uPSI_IdFingerServer;                //Indy
22000: 600 unit uPSI_IdDNSResolver;                 //Indy
22001: 601 unit uPSI_IdHTTPWebBrokerBridge;         //Indy
22002: 602 unit uPSI_IdIntercept;                   //Indy
22003: 603 unit uPSI_IdIPMCastBase;                 //Indy
22004: 604 unit uPSI_IdLogBase;                     //Indy
22005: 605 unit uPSI_IdIOHandlerStream;             //Indy
22006: 606 unit uPSI_IdMappedPortUDP;               //Indy
22007: 607 unit uPSI_IdQOTDUDPServer;               //Indy
22008: 608 unit uPSI_IdQOTDUDP;                     //Indy
22009: 609 unit uPSI_IdSysLog;                      //Indy
22010: 610 unit uPSI_IdSysLogServer;                //Indy
22011: 611 unit uPSI_IdSysLogMessage;               //Indy
22012: 612 unit uPSI_IdTimeServer;                  //Indy
22013: 613 unit uPSI_IdTimeUDP;                     //Indy
22014: 614 unit uPSI_IdTimeUDPServer;               //Indy
22015: 615 unit uPSI_IdUserAccounts;                //Indy
22016: 616 unit uPSI_TextUtils;                     //mX4
22017: 617 unit uPSI_MandelbrotEngine;              //mX4
22018: 618 unit uPSI_delphi_arduino_Unit1;          //mX4
22019: 619 unit uPSI_DTDSchema2;                    //mX4
22020: 620 unit uPSI_fplotMain;                     //DMath
22021: 621 unit uPSI_FindFileIter;                  //mX4
22022: 622 unit uPSI_PppState;  (JclStrHashMap)     //PPP
22023: 623 unit uPSI_PppParser;                     //PPP
22024: 624 unit uPSI_PppLexer;                      //PPP
22025: 625 unit uPSI_PCharUtils;                    //PPP
22026: 626 unit uPSI_uJSON;                         //WU
22027: 627 unit uPSI_JclStrHashMap;                 //JCL
22028: 628 unit uPSI_JclHookExcept;                 //JCL
```

```
22029: 629 unit uPSI_EncdDecd;                    //VCL
22030: 630 unit uPSI_SockAppReg;                 //VCL
22031: 631 unit uPSI_PJFileHandle;               //PJ
22032: 632 unit uPSI_PJEnvVars;                  //PJ
22033: 633 unit uPSI_PJPipe;                     //PJ
22034: 634 unit uPSI_PJPipeFilters;              //PJ
22035: 635 unit uPSI_PJConsoleApp;               //PJ
22036: 636 unit uPSI_UConsoleAppEx;              //PJ
22037: 637 unit uPSI_DbxSocketChannelNative,     //VCL
22038: 638 unit uPSI_DbxDataGenerator,           //VCL
22039: 639 unit uPSI_DBXClient;                  //VCL
22040: 640 unit uPSI_IdLogEvent;                 //Indy
22041: 641 unit uPSI_Reversi;                    //mX4
22042: 642 unit uPSI_Geometry;                   //mX4
22043: 643 unit uPSI_IdSMTPServer;               //Indy
22044: 644 unit uPSI_Textures;                   //mX4
22045: 645 unit uPSI_IBX;                        //VCL
22046: 646 unit uPSI_IWDBCommon;                 //VCL
22047: 647 unit uPSI_SortGrid;                   //mX4
22048: 648 unit uPSI_IB;                         //VCL
22049: 649 unit uPSI_IBScript;                   //VCL
22050: 650 unit uPSI_JvCSVBaseControls;          //JCL
22051: 651 unit uPSI_Jvg3DColors;                //JCL
22052: 652 unit uPSI_JvHLEditor;  //beat         //JCL
22053: 653 unit uPSI_JvShellHook;                //JCL
22054: 654 unit uPSI_DBCommon2                   //VCL
22055: 655 unit uPSI_JvSHFileOperation;          //JCL
22056: 656 unit uPSI_uFilexport;                 //mX4
22057: 657 unit uPSI_JvDialogs;                  //JCL
22058: 658 unit uPSI_JvDBTreeView;               //JCL
22059: 659 unit uPSI_JvDBUltimGrid;              //JCL
22060: 660 unit uPSI_JvDBQueryParamsForm;        //JCL
22061: 661 unit uPSI_JvExControls;               //JCL
22062: 662 unit uPSI_JvBDEMemTable;              //JCL
22063: 663 unit uPSI_JvCommStatus;               //JCL
22064: 664 unit uPSI_JvMailSlots2;               //JCL
22065: 665 unit uPSI_JvgWinMask;                 //JCL
22066: 666 unit uPSI_StEclpse;                   //SysTools
22067: 667 unit uPSI_StMime;                     //SysTools
22068: 668 unit uPSI_StList;                     //SysTools
22069: 669 unit uPSI_StMerge;                    //SysTools
22070: 670 unit uPSI_StStrS;                     //SysTools
22071: 671 unit uPSI_StTree,                     //SysTools
22072: 672 unit uPSI_StVArr;                     //SysTools
22073: 673 unit uPSI_StRegIni;                   //SysTools
22074: 674 unit uPSI_urkf;                       //DMath
22075: 675 unit uPSI_usvd;                       //DMath
22076: 676 unit uPSI_DepWalkUtils;               //JCL
22077: 677 unit uPSI_OptionsFrm;                 //JCL
22078: 678 unit yuvconverts;                     //mX4
22079: 679 uPSI_JvPropAutoSave;                  //JCL
22080: 680 uPSI_AclAPI;                          //alcinoe
22081: 681 uPSI_AviCap;                          //alcinoe
22082: 682 uPSI_ALAVLBinaryTree;                 //alcinoe
22083: 683 uPSI_ALFcnMisc;                       //alcinoe
22084: 684 uPSI_ALStringList;                    //alcinoe
22085: 685 uPSI_ALQuickSortList;                 //alcinoe
22086: 686 uPSI_ALStaticText;                    //alcinoe
22087: 687 uPSI_ALJSONDoc;                       //alcinoe
22088: 688 uPSI_ALGSMComm;                       //alcinoe
22089: 689 uPSI_ALWindows;                       //alcinoe
22090: 690 uPSI_ALMultiPartFormDataParser;       //alcinoe
22091: 691 uPSI_ALHttpCommon;                    //alcinoe
22092: 692 uPSI_ALWebSpider,                     //alcinoe
22093: 693 uPSI_ALHttpClient;                    //alcinoe
22094: 694 uPSI_ALFcnHTML;                       //alcinoe
22095: 695 uPSI_ALFTPClient;                     //alcinoe
22096: 696 uPSI_ALInternetMessageCommon;         //alcinoe
22097: 697 uPSI_ALWininetHttpClient;             //alcinoe
22098: 698 uPSI_ALWinInetFTPClient;              //alcinoe
22099: 699 uPSI_ALWinHttpWrapper;                //alcinoe
22100: 700 uPSI_ALWinHttpClient;                 //alcinoe
22101: 701 uPSI_ALFcnWinSock;                    //alcinoe
22102: 702 uPSI_ALFcnSQL;                        //alcinoe
22103: 703 uPSI_ALFcnCGI;                        //alcinoe
22104: 704 uPSI_ALFcnExecute;                    //alcinoe
22105: 705 uPSI_ALFcnFile;                       //alcinoe
22106: 706 uPSI_ALFcnMime;                       //alcinoe
22107: 707 uPSI_ALPhpRunner;                     //alcinoe
22108: 708 uPSI_ALGraphic;                       //alcinoe
22109: 709 uPSI_ALIniFiles;                      //alcinoe
22110: 710 uPSI_ALMemCachedClient;               //alcinoe
22111: 711 unit uPSI_MyGrids;                    //mX4
22112: 712 uPSI_ALMultiPartMixedParser           //alcinoe
22113: 713 uPSI_ALSMTPClient                     //alcinoe
22114: 714 uPSI_ALNNTPClient;                    //alcinoe
22115: 715 uPSI_ALHintBalloon;                   //alcinoe
22116: 716 unit uPSI_ALXmlDoc;                   //alcinoe
22117: 717 unit uPSI_IPCThrd;                    //VCL
```

```
22118: 718 unit uPSI_MonForm;                    //VCL
22119: 719 unit uPSI_TeCanvas;                   //Orpheus
22120: 720 unit uPSI_Ovcmisc;                    //Orpheus
22121: 721 unit uPSI_ovcfiler;                   //Orpheus
22122: 722 unit uPSI_ovcstate;                   //Orpheus
22123: 723 unit uPSI_ovccoco;                    //Orpheus
22124: 724 unit uPSI_ovcrvexp;                   //Orpheus
22125: 725 unit uPSI_OvcFormatSettings;          //Orpheus
22126: 726 unit uPSI_OvcUtils;                   //Orpheus
22127: 727 unit uPSI_ovcstore;                   //Orpheus
22128: 728 unit uPSI_ovcstr;                     //Orpheus
22129: 729 unit uPSI_ovcmru;                     //Orpheus
22130: 730 unit uPSI_ovccmd;                     //Orpheus
22131: 731 unit uPSI_ovctimer;                   //Orpheus
22132: 732 unit uPSI_ovcintl;                    //Orpheus
22133: 733 uPSI_AfCircularBuffer;                //AsyncFree
22134: 734 uPSI_AfUtils;                         //AsyncFree
22135: 735 uPSI_AfSafeSync;                      //AsyncFree
22136: 736 uPSI_AfComPortCore;                   //AsyncFree
22137: 737 uPSI_AfComPort;                       //AsyncFree
22138: 738 uPSI_AfPortControls;                  //AsyncFree
22139: 739 uPSI_AfDataDispatcher;                //AsyncFree
22140: 740 uPSI_AfViewers;                       //AsyncFree
22141: 741 uPSI_AfDataTerminal;                  //AsyncFree
22142: 742 uPSI_SimplePortMain;                  //AsyncFree
22143: 743 unit uPSI_ovcclock;                   //Orpheus
22144: 744 unit uPSI_o32intlst;                  //Orpheus
22145: 745 unit uPSI_o32ledlabel;                //Orpheus
22146: 746 unit uPSI_AlMySqlClient;              //alcinoe
22147: 747 unit uPSI_ALFBXClient;                //alcinoe
22148: 748 unit uPSI_ALFcnSQL;                   //alcinoe
22149: 749 unit uPSI_AsyncTimer;                 //mX4
22150: 750 unit uPSI_ApplicationFileIO;          //mX4
22151: 751 unit uPSI_PsAPI;                      //VCLé
22152: 752 uPSI_ovcuser;                         //Orpheus
22153: 753 uPSI_ovcurl;                          //Orpheus
22154: 754 uPSI_ovcvlb;                          //Orpheus
22155: 755 uPSI_ovccolor;                        //Orpheus
22156: 756 uPSI_ALFBXLib,                        //alcinoe
22157: 757 uPSI_ovcmeter;                        //Orpheus
22158: 758 uPSI_ovcpeakm;                        //Orpheus
22159: 759 uPSI_O32BGSty;                        //Orpheus
22160: 760 uPSI_ovcBidi;                         //Orpheus
22161: 761 uPSI_ovctcary;                        //Orpheus
22162: 762 uPSI_DXPUtils;                        //mX4
22163: 763 uPSI_ALMultiPartBaseParser;           //alcinoe
22164: 764 uPSI_ALMultiPartAlternativeParser;    //alcinoe
22165: 765 uPSI_ALPOP3Client;                    //alcinoe
22166: 766 uPSI_SmallUtils;                      //mX4
22167: 767 uPSI_MakeApp;                         //mX4
22168: 768 uPSI_O32MouseMon;                     //Orpheus
22169: 769 uPSI_OvcCache;                        //Orpheus
22170: 770 uPSI_ovccalc;                         //Orpheus
22171: 771 uPSI_Joystick,                        //OpenGL
22172: 772 uPSI_ScreenSaver;                     //OpenGL
22173: 773 uPSI_XCollection,                     //OpenGL
22174: 774 uPSI_Polynomials,                     //OpenGL
22175: 775 uPSI_PersistentClasses, //9.86       //OpenGL
22176: 776 uPSI_VectorLists;                     //OpenGL
22177: 777 uPSI_XOpenGL,                        //OpenGL
22178: 778 uPSI_MeshUtils;                      //OpenGL
22179: 779 unit uPSI_JclSysUtils;               //JCL
22180: 780 unit uPSI_JclBorlandTools;           //JCL
22181: 781 unit JclFileUtils_max;               //JCL
22182: 782 uPSI_AfDataControls,                 //AsyncFree
22183: 783 uPSI_GLSilhouette;                   //OpenGL
22184: 784 uPSI_JclSysUtils_class;              //JCL
22185: 785 uPSI_JclFileUtils_class;             //JCL
22186: 786 uPSI_FileUtil;                       //JCL
22187: 787 uPSI_changefind;                     //mX4
22188: 788 uPSI_cmdIntf;                        //mX4
22189: 789 uPSI_fservice;                       //mX4
22190: 790 uPSI_Keyboard;                       //OpenGL
22191: 791 uPSI_VRMLParser,                     //OpenGL
22192: 792 uPSI_GLFileVRML,                     //OpenGL
22193: 793 uPSI_Octree;                         //OpenGL
22194: 794 uPSI_GLPolyhedron,                   //OpenGL
22195: 795 uPSI_GLCrossPlatform;                //OpenGL
22196: 796 uPSI_GLParticles;                    //OpenGL
22197: 797 uPSI_GLNavigator;                    //OpenGL
22198: 798 uPSI_GLStarRecord;                   //OpenGL
22199: 799 uPSI_GLTextureCombiners;             //OpenGL
22200: 800 uPSI_GLCanvas;                       //OpenGL
22201: 801 uPSI_GeometryBB;                     //OpenGL
22202: 802 uPSI_GeometryCoordinates;            //OpenGL
22203: 803 uPSI_VectorGeometry;                 //OpenGL
22204: 804 uPSI_BumpMapping;                    //OpenGL
22205: 805 uPSI_TGA;                            //OpenGL
22206: 806 uPSI_GLVectorFileObjects;            //OpenGL
```

```
22207: 807 uPSI_IMM;                              //VCL
22208: 808 uPSI_CategoryButtons;                  //VCL
22209: 809 uPSI_ButtonGroup;                      //VCL
22210: 810 uPSI_DbExcept;                        //VCL
22211: 811 uPSI_AxCtrls;                         //VCL
22212: 812 uPSI_GL_actorUnit1;                   //OpenGL
22213: 813 uPSI_StdVCL;                         //VCL
22214: 814 unit CurvesAndSurfaces;               //OpenGL
22215: 815 uPSI_DataAwareMain;                   //AsyncFree
22216: 816 uPSI_TabNotBk;                       //VCL
22217: 817 uPSI_udwsfiler;                      //mX4
22218: 818 uPSI_synaip;                         //Synapse!
22219: 819 uPSI_synacode;                       //Synapse
22220: 820 uPSI_synachar;                       //Synapse
22221: 821 uPSI_synamisc;                       //Synapse
22222: 822 uPSI_synaser;                        //Synapse
22223: 823 uPSI_synaicnv;                       //Synapse
22224: 824 uPSI_tlntsend;                       //Synapse
22225: 825 uPSI_pingsend;                       //Synapse
22226: 826 uPSI_blcksock;                       //Synapse
22227: 827 uPSI_asn1util;                       //Synapse
22228: 828 uPSI_dnssend;                        //Synapse
22229: 829 uPSI_clamsend;                       //Synapse
22230: 830 uPSI_ldapsend;                       //Synapse
22231: 831 uPSI_mimemess;                       //Synapse
22232: 832 uPSI_slogsend;                       //Synapse
22233: 833 uPSI_mimepart;                       //Synapse
22234: 834 uPSI_mimeinln;                       //Synapse
22235: 835 uPSI_ftpsend,                        //Synapse
22236: 836 uPSI_ftptsend;                       //Synapse
22237: 837 uPSI_httpsend;                       //Synapse
22238: 838 uPSI_sntpsend;                       //Synapse
22239: 839 uPSI_smtpsend;                       //Synapse
22240: 840 uPSI_snmpsend;                       //Synapse
22241: 841 uPSI_imapsend;                       //Synapse
22242: 842 uPSI_pop3send;                       //Synapse
22243: 843 uPSI_nntpsend;                       //Synapse
22244: 844 uPSI_ssl_cryptlib;                   //Synapse
22245: 845 uPSI_ssl_openssl;                    //Synapse
22246: 846 uPSI_synhttp_daemon;                 //Synapse
22247: 847 uPSI_NetWork;                        //mX4
22248: 848 uPSI_PingThread;                     //Synapse
22249: 849 uPSI_JvThreadTimer;                  //JCL
22250: 850 unit uPSI_wwSystem;                  //InfoPower
22251: 851 unit uPSI_IdComponent;               //Indy
22252: 852 unit uPSI_IdIOHandlerThrottle;       //Indy
22253: 853 unit uPSI_Themes;                    //VCL
22254: 854 unit uPSI_StdStyleActnCtrls;         //VCL
22255: 855 unit uPSI_UDDIHelper;                //VCL
22256: 856 unit uPSI_IdIMAP4Server;             //Indy
22257: 857 uPSI_VariantSymbolTable,             //VCL //3.9.9.92
22258: 858 uPSI_udf_glob,                       //mX4
22259: 859 uPSI_TabGrid,                        //VCL
22260: 860 uPSI_JsDBTreeView,                   //mX4
22261: 861 uPSI_JsSendMail,                     //mX4
22262: 862 uPSI_dbTvRecordList,                 //mX4
22263: 863 uPSI_TreeVwEx,                       //mX4
22264: 864 uPSI_ECDataLink,                     //mX4
22265: 865 uPSI_dbTree,                         //mX4
22266: 866 uPSI_dbTreeCBox,                     //mX4
22267: 867 unit uPSI_Debug; //TfrmDebug        //mX4
22268: 868 uPSI_TimeFncs;                       //mX4
22269: 869 uPSI_FileIntf,                      //VCL
22270: 870 uPSI_SockTransport,                 //RTL
22271: 871 unit uPSI_WinInet;                  //RTL
22272: 872 unit uPSI_Wwstr;                    //mX4
22273: 873 uPSI_DBLookup,                      //VCL
22274: 874 uPSI_Hotspot,                       //mX4
22275: 875 uPSI_HList; //History List          //mX4
22276: 876 unit uPSI_DrTable;                  //VCL
22277: 877 uPSI_TConnect,                      //VCL
22278: 878 uPSI_DataBkr,                       //VCL
22279: 879 uPSI_HTTPIntr;                      //VCL
22280: 880 unit uPSI_Mathbox;                  //mX4
22281: 881 uPSI_cyIndy,                        //cY
22282: 882 uPSI_cySysUtils,                    //cY
22283: 883 uPSI_cyWinUtils,                    //cY
22284: 884 uPSI_cyStrUtils,                    //cY
22285: 885 uPSI_cyObjUtils,                    //cY
22286: 886 uPSI_cyDateUtils,                   //cY
22287: 887 uPSI_cyBDE,                         //cY
22288: 888 uPSI_cyClasses,                     //cY
22289: 889 uPSI_cyGraphics,  //3.9.9.94_2      //cY
22290: 890 unit uPSI_cyTypes;                  //cY
22291: 891 uPSI_JvDateTimePicker,              //JCL
22292: 892 uPSI_JvCreateProcess,               //JCL
22293: 893 uPSI_JvEasterEgg,                   //JCL
22294: 894 uPSI_WinSvc,  //3.9.9.94_3          //VCL
22295: 895 uPSI_SvcMgr                         //VCL
```

```
22296: 896 unit uPSI_JvPickDate;                    //JCL
22297: 897 unit uPSI_JvNotify;                      //JCL
22298: 898 uPSI_JvStrHlder                          //JCL
22299: 899 unit uPSI_JclNTFS2;                      //JCL
22300: 900 uPSI_Jcl8087 //math coprocessor          //JCL
22301: 901 uPSI_JvAddPrinter                        //JCL
22302: 902 uPSI_JvCabFile                           //JCL
22303: 903 uPSI_JvDataEmbedded;                     //JCL
22304: 904 unit uPSI_U_HexView;                     //mX4
22305: 905 uPSI_UWavein4,                           //mX4
22306: 906 uPSI_AMixer,                             //mX4
22307: 907 uPSI_JvaScrollText,                      //mX4
22308: 908 uPSI_JvArrow,                           //mX4
22309: 909 unit uPSI_UrlMon;                       //mX4
22310: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
22311: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain; //DFF
22312: 912 unit uPSI_DFFUtils;                     //DFF
22313: 913 unit uPSI_MathsLib;                     //DFF
22314: 914 uPSI_UIntList;                          //DFF
22315: 915 uPSI_UGetParens;                        //DFF
22316: 916 unit uPSI_UGeometry;                    //DFF
22317: 917 unit uPSI_UAstronomy;                   //DFF
22318: 918 unit uPSI_UCardComponentV2;             //DFF
22319: 919 unit uPSI_UTGraphSearch;                //DFF
22320: 920 unit uPSI_UParser10;                    //DFF
22321: 921 unit uPSI_cyIEUtils;                    //cY
22322: 922 unit uPSI_UcomboV2;                     //DFF
22323: 923 uPSI_cyBaseComm,                        //cY
22324: 924 uPSI_cyAppInstances,                    //cY
22325: 925 uPSI_cyAttract,                         //cY
22326: 926 uPSI_cyDERUtils                         //cY
22327: 927 unit uPSI_cyDocER;                      //cY
22328: 928 unit uPSI_ODBC;                         //mX
22329: 929 unit uPSI_AssocExec;                    //mX
22330: 930 uPSI_cyBaseCommRoomConnector,           //cY
22331: 931 uPSI_cyCommRoomConnector,               //cY
22332: 932 uPSI_cyCommunicate,                     //cY
22333: 933 uPSI_cyImage;                           //cY
22334: 934 uPSI_cyBaseContainer                    //cY
22335: 935 uPSI_cyModalContainer,                  //cY
22336: 936 uPSI_cyFlyingContainer;                 //cY
22337: 937 uPSI_RegStr,                            //VCL
22338: 938 uPSI_HtmlHelpViewer;                    //VCL
22339: 939 unit uPSI_cyIniForm                     //cY
22340: 940 unit uPSI_cyVirtualGrid;                //cY
22341: 941 uPSI_Profiler,                          //DA
22342: 942 uPSI_BackgroundWorker,                  //DA
22343: 943 uPSI_WavePlay,                          //DA
22344: 944 uPSI_WaveTimer,                         //DA
22345: 945 uPSI_WaveUtils;                         //DA
22346: 946 uPSI_NamedPipes,                        //TB
22347: 947 uPSI_NamedPipeServer,                   //TB
22348: 948 unit uPSI_process,                      //TB
22349: 949 unit uPSI_DPUtils;                      //TB
22350: 950 unit uPSI_CommonTools;                  //TB
22351: 951 uPSI_DataSendToWeb,                     //TB
22352: 952 uPSI_StarCalc,                          //TB
22353: 953 uPSI_D2_XPVistaHelperU                  //TB
22354: 954 unit uPSI_NetTools                      //TB
22355: 955 unit uPSI_Pipes                         //TB
22356: 956 uPSI_ProcessUnit,                       //mX
22357: 957 uPSI_adGSM,                             //mX
22358: 958 unit uPSI_BetterADODataSet;             //mX
22359: 959 unit uPSI_AdSelCom; //FTT               //mX
22360: 960 unit unit uPSI_dwsXPlatform;            //DWS
22361: 961 uPSI_AdSocket;                          //mX Turbo Power
22362: 962 uPSI_AdPacket;                          //mX
22363: 963 uPSI_AdPort;                            //mX
22364: 964 uPSI_PathFunc;                          //Inno
22365: 965 uPSI_CmnFunc;                           //Inno
22366: 966 uPSI_CmnFunc2; //Inno Setup             //Inno
22367: 967 unit uPSI_BitmapImage;                  //mX4
22368: 968 unit uPSI_ImageGrabber;                 //mX4
22369: 969 uPSI_SecurityFunc,                      //Inno
22370: 970 uPSI_RedirFunc,                         //Inno
22371: 971 uPSI_FIFO, (MemoryStream)               //mX4
22372: 972 uPSI_Int64Em,                           //Inno
22373: 973 unit uPSI_InstFunc;                     //Inno
22374: 974 unit uPSI_LibFusion;                    //Inno
22375: 975 uPSI_SimpleExpression;                  //Inno
22376: 976 uPSI_unitResourceDetails,               //XN
22377: 977 uPSI_unitResFile,                       //XN
22378: 978 unit uPSI_simpleComport;                //mX4
22379: 979 unit uPSI_AfViewershelpers;             //Async
22380: 980 unit uPSI_Console;                      //mX4
22381: 981 unit uPSI_AnalogMeter;                  //TB
22382: 982 unit uPSI_XPrinter,                     //TB
22383: 983 unit uPSI_IniFiles;                     //VCL
22384: 984 unit uPSI_lazIniFiles;                  //FP
```

```
22385: 985 uPSI_testutils;                          //FP
22386: 986 uPSI_ToolsUnit; (DBTests)                //FP
22387: 987 uPSI_fpcunit                             //FP
22388: 988 uPSI_testdecorator;                      //FP
22389: 989 unit uPSI_fpcunittests;                  //FP
22390: 990 unit uPSI_cTCPBuffer;                    //Fundamentals 4
22391: 991 unit uPSI_Glut,                          //Open GL
22392: 992 uPSI_LEDBitmaps,                        //mX4
22393: 993 uPSI_FileClass,                         //Inno
22394: 994 uPSI_FileUtilsClass,                    //mX4
22395: 995 uPSI_ComPortInterface; //Kit           //mX4
22396: 996 unit uPSI_SwitchLed;                    //mX4
22397: 997 unit uPSI_cyDmmCanvas;                  //cY
22398: 998 uPSI_uColorFunctions;                   //DFF
22399: 999 uPSI_uSettings;                         //DFF
22400: 1000 uPSI_cyDebug.pas                       //cY
22401: 1001 uPSI_cyColorMatrix;                    //cY
22402: 1002 unit uPSI_cyCopyFiles;                 //cY
22403: 1003 unit uPSI_cySearchFiles;               //cY
22404: 1004 unit uPSI_cyBaseMeasure;               //cY
22405: 1005 unit uPSI_PJIStreams;                  //DD
22406: 1006 unit uPSI_cyRunTimeResize;             //cY
22407: 1007 unit uPSI_jcontrolutils;               //cY
22408: 1008 unit uPSI_kcMapViewer; (+GEONames)     //kc
22409: 1009 unit uPSI_kcMapViewerDESynapse;        //kc
22410: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
22411: 1011 unit uPSI_LedNumber;                   //TurboPower
22412: 1012 unit uPSI_StStrL;                      //SysTools
22413: 1013 unit uPSI_indGnouMeter;                //LAZ
22414: 1014 unit uPSI_Sensors;                     //LAZ
22415: 1015 unit uPSI_pwmain;  //cgi of powtils    //Pow
22416: 1016 unit uPSI_HTMLUtil;                    //Pow
22417: 1017 unit uPSI_synwrap1; //httpsend        //Pow
22418: 1018 unit StreamWrap1                       //Pow
22419: 1019 unit uPSI_pwmain;                      //Pow
22420: 1020 unit pwtypes                          //Pow
22421:
22422:
22423:
22424: ////////////////////////////////////////////////////////////////////////////
22425: //Form Template Library FTL
22426: ////////////////////////////////////////////////////////////////////////////
22427:
22428: 32 FTL For Form Building out of the Script, eg. 399_form_templates.txt
22429:
22430: 045 unit uPSI_VListView                      TFormListView;
22431: 263 unit uPSI_JvProfiler32;                  TProfReport
22432: 270 unit uPSI_ImgList;                       TCustomImageList
22433: 278 unit uPSI_JvImageWindow;                 TJvImageWindow
22434: 317 unit uPSI_JvParserForm;                  TJvHTMLParserForm
22435: 497 unit uPSI_DebugBox;                      TDebugBox
22436: 513 unit uPSI_ImageWin;                      TImageForm, TImageForm2
22437: 514 unit uPSI_CustomDrawTreeView;           TCustomDrawForm
22438: 515 unit uPSI_GraphWin;                     TGraphWinForm
22439: 516 unit uPSI_actionMain;                   TActionForm
22440: 518 unit uPSI_CtlPanel;                     TAppletApplication
22441: 529 unit uPSI_MDIEdit;                      TEditForm
22442: 535 unit uPSI_CoolMain; {browser}          TWebMainForm
22443: 538 unit uPSI_frmExportMain;               TSynexportForm
22444: 585 unit uPSI_usniffer; {//PortScanForm}   TSniffForm
22445: 600 unit uPSI_ThreadForm;                  TThreadSortForm;
22446: 618 unit uPSI_delphi_arduino_Unit1;        TLEDForm
22447: 620 unit uPSI_fplotMain;                   TfplotForm1
22448: 660 unit uPSI_JvDBQueryParamsForm;         TJvQueryParamsDialog
22449: 677 unit uPSI_OptionsFrm;                  TfrmOptions;
22450: 718 unit uPSI_MonForm;                     TMonitorForm
22451: 742 unit uPSI_SimplePortMain;              TPortForm1
22452: 770 unit uPSI_ovccalc;                     TOvcCalculator  //widget
22453: 810 unit uPSI_DbExcept;                    TDbEngineErrorDlg
22454: 812 unit uPSI_GL_actorUnit1;               TglActorForm1  //OpenGL Robot
22455: 846 unit uPSI_synhttp_daemon;              TTCPHttpDaemon, TTCPHttpThrd, TPingThread
22456: 867 unit uPSI_Debug;                       TfrmDebug
22457: 904 unit uPSI_U_HexView;                   THexForm2
22458: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)  TOscfrmMain
22459: 959 unit uPSI_AdSelCom;                    TComSelectForm
22460:
22461:
22462: ex.:with TEditForm.create(self) do begin
22463:       caption:= 'Template Form Tester';
22464:       FormStyle:= fsStayOnTop;
22465:       with editor do begin
22466:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
22467:         SelStart:= 0;
22468:         Modified:= False;
22469:       end;
22470:     end;
22471:   with TWebMainForm.create(self) do begin
22472:     URLs.Text:= 'http://www.kleiner.ch';
22473:     URLsClick(self); Show;
```

```
22474:   end;
22475:   with TSynexportForm.create(self) do begin
22476:     Caption:= 'Synexport HTML RTF tester';
22477:     Show;
22478:   end;
22479:   with TThreadSortForm.create(self) do begin
22480:      showmodal; free;
22481:   end;
22482:   with TCustomDrawForm.create(self) do begin
22483:        width:=820; height:=820;
22484:        image1.height:= 600; //add properties
22485:        image1.picture.bitmap:= image2.picture.bitmap;
22486:        //SelectionBackground1Click(self) CustomDraw1Click(self);
22487:        Background1.click;
22488:        bitmap1.click;
22489:        Tile1.click;
22490:        Showmodal;
22491:        Free;
22492:      end;
22493:    with TfplotForm1.Create(self) do begin
22494:      BtnPlotClick(self);
22495:      Showmodal; Free;
22496:    end;
22497:   with TOvcCalculator.create(self) do begin
22498:     parent:= aForm;
22499:    //free;
22500:     setbounds(550,510,200,150);
22501:     displaystr:= 'maXcalc';
22502:   end;
22503:   with THexForm2.Create(self) do begin
22504:      ShowModal;
22505:      Free;
22506:   end;
22507:
22508: function CheckBox: string;
22509: var idHTTP: TIDHTTP;
22510:  begin
22511:    result:= 'version not found';
22512:    if IsInternet then begin
22513:      idHTTP:= TIdHTTP.Create(NIL);
22514:      try
22515:        result:= idHTTP.Get(MXVERSIONFILE2);
22516:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
22517:        if result = MBVER2 then begin
22518:          //output.Font.Style:= [fsbold];
22519:          //Speak(' A new Version '+vstr+' of max box is available! ');
22520:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
22521:        end;
22522:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
22523:      finally
22524:        idHTTP.Free
22525:    end;
22526:  end;
22527: end;
22528:
22529:
22530: ////////////////////////////////////////////////////////////////////////////
22531: All maXbox Tutorials Table of Content 2014
22532: ////////////////////////////////////////////////////////////////////////////
22533:     Tutorial 00 Function-Coding  (Blix the Programmer)
22534:     Tutorial 01 Procedural-Coding
22535:     Tutorial 02 OO-Programming
22536:     Tutorial 03 Modular Coding
22537:     Tutorial 04 UML Use Case Coding
22538:     Tutorial 05 Internet Coding
22539:     Tutorial 06 Network Coding
22540:     Tutorial 07 Game Graphics Coding
22541:     Tutorial 08 Operating System Coding
22542:     Tutorial 09 Database Coding
22543:     Tutorial 10 Statistic Coding
22544:     Tutorial 11 Forms Coding
22545:     Tutorial 12 SQL DB Coding
22546:     Tutorial 13 Crypto Coding
22547:     Tutorial 14 Parallel Coding
22548:     Tutorial 15 Serial RS232 Coding
22549:     Tutorial 16 Event Driven Coding
22550:     Tutorial 17 Web Server Coding
22551:     Tutorial 18 Arduino System Coding
22552:     Tutorial 18_3 RGB LED System Coding
22553:     Tutorial 19 WinCOM /Arduino Coding
22554:     Tutorial 20 Regular Expressions RegEx
22555:     Tutorial 21 Android Coding (coming 2013)
22556:     Tutorial 22 Services Programming
22557:     Tutorial 23 Real Time Systems
22558:     Tutorial 24 Clean Code
22559:     Tutorial 25 maXbox Configuration I+II
22560:     Tutorial 26 Socket Programming with TCP
22561:     Tutorial 27 XML & TreeView
22562:     Tutorial 28 DLL Coding (available)
```

```
22563:     Tutorial 29 UML Scripting (2014)
22564:     Tutorial 30 Web of Things (2014)
22565:     Tutorial 31 Closures (2014)
22566:     Tutorial 32 SQL Firebird (coming 2014)
22567:     Tutorial 33 Oscilloscope (coming 2015)
22568:     Tutorial 34 GPS Navigation (2014)
22569:     Tutorial 35 Web Box (available)
22570:     Tutorial 36 Unit Testing (coming 2015)
22571:     Tutorial 37 API Coding (coming 2015)
22572:     Tutorial 38 3D Coding (coming 2015)
22573:     Tutorial 39 GEO Map Coding (available)
22574:     Tutorial 39_1 GEO Map Layers Coding (available)
22575:     Tutorial 40 REST Coding (coming 2015)
22576:
22577:
22578: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
22579: using Docu for this file is maxbox_functions_all.pdf
22580: PEP - Pascal Education Program  Low Lib Lab ShellHell
22581:
22582:
22583: http://stackoverflow.com/tags/pascalscript/hot
22584: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
22585: http://sourceforge.net/projects/maXbox   #locs:51620
22586: http://sourceforge.net/apps/mediawiki/maXbox
22587: http://www.blaisepascal.eu/
22588: https://github.com/maxkleiner/maXbox3.git
22589: http://www.heise.de/download/maxbox-1176464.html
22590: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
22591: https://www.facebook.com/pages/Programming-maXbox/166844836691703
22592: http://www.softwareschule.ch/arduino_training.pdf
22593: http://www.delphiarea.com
22594: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
22595:
22596:
22597: All maXbox Examples List
22598: https://github.com/maxkleiner/maXbox3/releases
22599: ********************************************************************************************
22600: 000_pas_baseconvert.txt                          282_fadengraphik.txt
22601: 000_pas_baseconvert.txt_encrypt                  283_SQL_API_messagetimeout.txt
22602: 000_pas_baseconvert.txt_decrypt                  284_SysTools4.txt
22603: 001_1_pas_functest - Kopie.txt                   285_MineForm_GR32.TXT
22604: 001_1_pas_functest.txt                          285_MineForm_GR32main.TXT
22605: 001_1_pas_functest2.txt                         285_MineForm_GR32mainsolution.TXT
22606: 001_1_pas_functest_clx2.txt                    285_MineForm_propas.TXT
22607: 001_1_pas_functest_clx2_2.txt                  285_MineForm_propas2.TXT
22608: 001_1_pas_functest_openarray.txt               285_minesweeper2.TXT
22609: 001_pas_lottogen.txt                           285_Patterns_process.txt
22610: 001_pas_lottogen_template.txt                  286_colormixer_jpeg_charcounter.txt
22611: 001_pas_lottogen.txtcopy                       286_colormixer_jpeg_charcounter2.txt
22612: 002_pas_russianroulette.txt                    287_eventhandling.txt
22613: 002_pas_russianroulette.txtcopy                287_eventhandling2.txt
22614: 002_pas_russianroulette.txtcopy_decrypt        287_eventhandling2_negpower.txt
22615: 002_pas_russianroulette.txtcopy_encrypt        288_bitblt.txt
22616: 003_pas_motion.txt                             288_bitblt_resize.txt
22617: 003_pas_motion.txtcopy                         289_regression.txt
22618: 004_pas_search.txt                             289_regression2.txt
22619: 004_pas_search_replace.txt                     290_bestofbox.txt
22620: 004_search_replace_allfunctionlist.txt         290_bestofbox2.txt
22621: 005_pas_oodesign.txt                           290_bestofbox3.txt
22622: 005_pas_shelllink.txt                          291_3sort_visual_thread.txt
22623: 006_pas_oobatch.txt                            292_refactoring2.txt
22624: 007_pas_streamcopy.txt                         293_bold_utils.txt
22625: 008_EINMALEINS_FUNC.TXT                        293_ib_utils.txt
22626: 008_explanation.txt                            293_ib_utils_timetest.txt
22627: 008_pas_verwechselt.txt                        294_maxcalc_demo.txt
22628: 008_pas_verwechselt_ibz_bern_func.txt          294_maxcalc_demo2.txt
22629: 008_stack_ibz.TXT                              295_easter_calendar.txt
22630: 009_pas_umlrunner.txt                          295_easter_calendar2.txt
22631: 009_pas_umlrunner_all.txt                      295_easter_combobox.txt
22632: 009_pas_umlrunner_componenttest.txt            297_atomimage.txt
22633: 009_pas_umlrunner_solution.txt                 297_atomimage2.txt
22634: 009_pas_umlrunner_solution_2step.txt           297_atomimage3.txt
22635: 010_pas_oodesign_solution.txt                  297_atomimage4.txt
22636: 011_pas_puzzlepas_defect.txt                   297_maxonmotor.TXT
22637: 012_pas_umlrunner_solution.txt                 297_maxon_atomimage9.txt
22638: 012_pas_umlrunner_solution2.txt                298_bitblt_animation.txt
22639: 013_pas_linenumber.txt                         298_bitblt_animation2.txt
22640: 014_pas_primetest.txt                          298_bitblt_animation3.txt
22641: 014_pas_primetest_first.txt                    298_bitblt_animation4.txt
22642: 014_pas_primetest_sync.txt                     298_bitblt_animation4_screensaver.txt
22643: 015_pas_designbycontract.txt                   298_bitblt_animation5_screensaver.txt
22644: 015_pas_designbycontract_solution.txt          299_animation.txt
22645: 016_pas_searchrec.txt                          299_animationmotor_arduino.txt
22646: 017_chartgen.txt                               299_animation_formprototype.txt
22647: 018_data_simulator.txt                         299_realtimeclock_arduino.txt
22648: 019_dez_to_bin.txt                             299_realtimeclock_arduino2.txt
22649: 019_dez_to_bin_grenzwert_ibz.txt               300_treeview.txt
22650: 020_proc_feedback.txt                          300_treeview_test.txt
22651: 021_pas_symkey.txt                             300_treeview_test2.txt
```

22652: 021_pas_symkey_solution.txt
22653: 022_pas_filestreams.txt
22654: 023_pas_find_searchrec.txt
22655: 023_pas_pathfind.txt
22656: 024_pas_TFileStream_records.txt
22657: 025_prime_direct.txt
22658: 026_pas_memorystream.txt
22659: 027_pas_shellexecute_beta.txt
22660: 027_pas_shellexecute_solution.txt
22661: 028_pas_dataset.txt
22662: 029_pas_assignfile.txt
22663: 029_pas_assignfile_dragndropexe.txt
22664: 030_palindrome_2.txt
22665: 030_palindrome_tester.txt
22666: 030_pas_recursion.txt
22667: 030_pas_recursion2.txt
22668: 031_pas_hashcode.txt
22669: 032_pas_crc_const.txt
22670: 033_pas_cipher.txt
22671: 033_pas_cipher_def.txt
22672: 033_pas_cipher_file_2_solution.txt
22673: 034_pas_soundbox.txt
22674: 035_pas_crcscript.txt
22675: 035_pas_CRCscript_modbus.txt
22676: 036_pas_includetest.txt
22677: 036_pas_includetest_basta.txt
22678: 037_pas_define_demo32.txt
22679: 038_pas_box_demonstrator.txt
22680: 039_pas_dllcall.txt
22681: 040_paspointer.txt
22682: 040_paspointer_old.txt
22683: 041_pasplotter.txt
22684: 041_pasplotter_plus.txt
22685: 042_pas_kgv_ggt.txt
22686: 043_pas_proceduretype.txt
22687: 044_pas_14queens_solwith14.txt
22688: 044_pas_8queens.txt
22689: 044_pas_8queens_sol2.txt
22690: 044_pas_8queens_solutions.txt
22691: 044_queens_performer.txt
22692: 044_queens_performer2.txt
22693: 044_queens_performer2tester.txt
22694: 045_pas_listhandling.txt
22695: 046_pas_records.txt
22696: 047_pas_modula10.txt
22697: 048_pas_romans.txt
22698: 049_pas_ifdemo.txt
22699: 049_pas_ifdemo_BROKER.txt
22700: 050_pas_primetest2.txt
22701: 050_pas_primetester_thieves.txt
22702: 050_program_starter.txt
22703: 050_program_starter_performance.txt
22704: 051_pas_findtext_solution.txt
22705: 052_pas_text_as_stream.txt
22706: 052_pas_text_as_stream_include.txt
22707: 053_pas_singleton.txt
22708: 054_pas_speakpassword.txt
22709: 054_pas_speakpassword2.txt
22710: 054_pas_speakpassword_searchtest.txt
22711: 055_pas_factorylist.txt
22712: 056_pas_demeter.txt
22713: 057_pas_dirfinder.txt
22714: 058_pas_filefinder.txt
22715: 058_pas_filefinder_pdf.txt
22716: 058_pas_filefinder_screview.txt
22717: 058_pas_filefinder_screview2.txt
22718: 058_pas_filefinder_screview3.txt
22719: 059_pas_timertest.txt
22720: 059_pas_timertest_2.txt
22721: 059_pas_timertest_time_solution.txt
22722: 059_timerobject_starter2.txt
22723: 059_timerobject_starter2_ibz2_async.txt
22724: 059_timerobject_starter2_uml.txt
22725: 059_timerobject_starter2_uml_main.txt
22726: 059_timerobject_starter4_ibz.txt
22727: 060_pas_datefind.txt
22728: 060_pas_datefind_exceptions2.txt
22729: 060_pas_datefind_exceptions_CHECKTEST.txt
22730: 060_pas_datefind_fulltext.txt
22731: 060_pas_datefind_plus.txt
22732: 060_pas_datefind_plus_mydate.txt
22733: 061_pas_randomwalk.txt
22734: 061_pas_randomwalk_plus.txt
22735: 062_paskorrelation.txt
22736: 063_pas_calculateform.txt
22737: 063_pas_calculateform_2list.txt
22738: 064_pas_timetest.txt
22739: 065_pas_bitcounter.txt
22740: 066_pas_eliza.txt

300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT
315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
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

22741: 066_pas_eliza_include_sol.txt
22742: 067_pas_morse.txt
22743: 068_pas_piezo_sound.txt
22744: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
22745: 069_my_LEDBOX.TXT
22746: 069_pas_ledmatrix.txt
22747: 069_pas_LEDMATRIX_Alphabet.txt
22748: 069_pas_LEDMATRIX_Alphabet_run.txt
22749: 069_pas_LEDMATRIX_Alphabet_tester.txt
22750: 069_PAS_LEDMATRIX_COLOR.TXT
22751: 069_pas_ledmatrix_fixedit.txt
22752: 069_pas_LEDMATRIX_soundbox.txt
22753: 069_pas_LEDMATRIX_soundbox2.txt
22754: 069_Richter_MATRIX.TXT
22755: 070_pas_functionplot.txt
22756: 070_pas_functionplotter2.txt
22757: 070_pas_functionplotter2_mx4.txt
22758: 070_pas_functionplotter2_tester.txt
22759: 070_pas_functionplotter3.txt
22760: 070_pas_functionplotter4.txt
22761: 070_pas_functionplotter_digital.txt
22762: 070_pas_functionplotter_elliptic.txt
22763: 070_pas_function_helmholtz.txt
22764: 070_pas_textcheck_experimental.txt
22765: 071_pas_graphics.txt
22766: 071_pas_graphics_drawsym.txt
22767: 071_pas_graphics_drawsym_save.txt
22768: 071_pas_graphics_random.txt
22769: 072_pas_fractals.txt
22770: 072_pas_fractals_2.txt
22771: 072_pas_fractals_blackhole.txt
22772: 072_pas_fractals_perfomance.txt
22773: 072_pas_fractals_perfomance_new.txt
22774: 072_pas_fractals_perfomance_sharp.txt
22775: 072_pas_fractals_performance.txt
22776: 072_pas_fractals_performance_mX4.txt
22777: 073_pas_forms.txt
22778: 074_pas_chartgenerator.txt
22779: 074_pas_chartgenerator_solution.txt
22780: 074_pas_chartgenerator_solution_back.txt
22781: 074_pas_charts.txt
22782: 075_bitmap_Artwork2.txt
22783: 075_pas_bitmappuzzle.txt
22784: 075_pas_bitmappuzzle24.prod.txt
22785: 075_pas_bitmappuzzle2_prod.txt
22786: 075_pas_bitmappuzzle3.txt
22787: 075_pas_bitmapsolve.txt
22788: 075_pas_bitmap_Artwork.txt
22789: 075_pas_puzzlepas_solution.txt
22790: 076_pas_3dcube.txt
22791: 076_pas_circle.txt
22792: 077_pas_mmshow.txt
22793: 078_pas_pi.txt
22794: 079_pas_3dcube_animation.txt
22795: 079_pas_3dcube_animation4.txt
22796: 079_pas_3dcube_plus.txt
22797: 080_pas_hanoi.txt
22798: 080_pas_hanoi2.txt
22799: 080_pas_hanoi2_file.txt
22800: 080_pas_hanoi2_sol.txt
22801: 080_pas_hanoi2_tester.txt
22802: 080_pas_hanoi2_tester_fast.txt
22803: 080_pas_hanoi3.txt
22804: 081_pas_chartist2.txt
22805: 082_pas_biorythmus.txt
22806: 082_pas_biorythmus_solution.txt
22807: 082_pas_biorythmus_solution_3.txt
22808: 082_pas_biorythmus_test.txt
22809: 083_pas_GITARRE.txt
22810: 083_pas_soundbox_tones.txt
22811: 084_pas_waves.txt
22812: 085_mxsinus_logo.txt
22813: 085_sinus_plot_waves.txt
22814: 086_pas_graph_arrow_heart.txt
22815: 087_bitmap_loader.txt
22816: 087_pas_bitmap_solution.txt
22817: 087_pas_bitmap_solution2.txt
22818: 087_pas_bitmap_subimage.txt
22819: 087_pas_bitmap_test.txt
22820: 088_pas_soundbox2_mp3.txt
22821: 088_pas_soundbox_mp3.txt
22822: 088_pas_sphere_2.txt
22823: 089_pas_gradient.txt
22824: 089_pas_maxland2.txt
22825: 090_pas_sudoku4.txt
22826: 090_pas_sudoku4_2.txt
22827: 091_pas_cube4.txt
22828: 092_pas_statistics4.txt
22829: 093_variance.txt

337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docutype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_biginteger.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT
362_maxonmotor2.TXT
363_compress_services.txt
363_compress_services2.txt
364_pdf_services.txt
365_memorystream.txt
365_memorystream2.txt
365_memorystream_test.txt
365_U_HexView.txt
366_mp3player.txt
366_mp3player2.txt
366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maXbook_v4tester.txt
372stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
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

22830: 093_variance_debug.txt
22831: 094_pas_daysold.txt
22832: 094_pas_stat_date.txt
22833: 095_pas_ki_simulation.txt
22834: 096_pas_geisen_problem.txt
22835: 096_pas_montyhall_problem.txt
22836: 097_lotto_proofofconcept.txt
22837: 097_pas_lottocombinations_beat_plus.txt
22838: 097_pas_lottocombinations_beat_plus2.txt
22839: 097_pas_lottocombinations_universal.txt
22840: 097_pas_lottosimulation.txt
22841: 098_pas_chartgenerator_plus.txt
22842: 099_pas_3D_show.txt
22843: 200_big_numbers.txt
22844: 200_big_numbers2.txt
22845: 201_streamload_xml.txt
22846: 202_systemcheck.txt
22847: 203_webservice_simple_intftester.txt
22848: 204_webservice_simple.txt
22849: 205_future_value_service.txt
22850: 206_DTD_string_functions.txt
22851: 207_ibz2_async_process.txt
22852: 208_crc32_hash.txt
22853: 209_cryptohash.txt
22854: 210_public_private.txt
22855: 210_public_private_cryptosystem.txt
22856: 211_wipe_pattern.txt
22857: 211_wipe_pattern2.txt
22858: 211_wipe_pattern_solution.txt
22859: 212_pas_statisticmodule4.TXT
22860: 212_pas_statisticmoduletxt.TXT
22861: 212_statisticmodule4.txt
22862: 213_pas_BBP_Algo.txt
22863: 214_mxdocudemo.txt
22864: 214_mxdocudemo2.txt
22865: 214_mxdocudemo3.txt
22866: 215_hints_test.TXT
22867: 216_warnings_test.TXT
22868: 217_pas_heartbeat.txt
22869: 218_biorythmus_studio.txt
22870: 219_cipherbox.txt
22871: 219_crypt_source_comtest_solution.TXT
22872: 220_cipherbox_form.txt
22873: 220_cipherbox_form2.txt
22874: 221_bcd_explain.txt
22875: 222_memoform.txt
22876: 223_directorybox.txt
22877: 224_dialogs.txt
22878: 225_sprite_animation.txt
22879: 226_ASCII_Grid2.TXT
22880: 227_animation.txt
22881: 227_animation2.txt
22882: 228_android_calendar.txt
22883: 229_android_game.txt
22884: 229_android_game_tester.txt
22885: 230_DataProvider.txt
22886: 230_DataSetProvider.txt
22887: 230_DataSetXMLBackupScholz.txt
22888: 231_DBGrid_access.txt
22889: 231_DBGrid_XMLaccess.txt
22890: 231_DBGrid_XMLaccess2.txt
22891: 231_DBGrid_XMLaccess_locatetester.txt
22892: 231_DBGrid_XML_CDS_local.txt
22893: 232_outline.txt
22894: 232_outline_2.txt
22895: 233_modular_form.txt
22896: 234_debugoutform.txt
22897: 235_fastform.TXT
22898: 236_componentpower.txt
22899: 236_componentpower_back.txt
22900: 237_pas_4forms.txt
22901: 238_lottogen_form.txt
22902: 239_pas_sierpinski.txt
22903: 239_pas_sierpinski2.txt
22904: 240_unitGlobal_tester.txt
22905: 241_db3_sql_tutorial.txt
22906: 241_db3_sql_tutorial2.txt
22907: 241_db3_sql_tutorial2fix.txt
22908: 241_db3_sql_tutorial3.txt
22909: 241_db3_sql_tutorial3connect.txt
22910: 241_db3_sql_tutorial3_fptest.txt
22911: 241_RTL_SET2.txt
22912: 241_RTL_SET2_tester.txt
22913: 242_Component_Control.txt
22914: 243_tutorial_loader.txt
22915: 244_script_loader_loop.txt
22916: 245_formapp2.txt
22917: 245_formapp2_tester.txt
22918: 245_formapp2_testerX.txt

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

22919: 246_httpapp.txt
22920: 247_datecalendar.txt
22921: 248_ASCII_Grid2_sorted.TXT
22922: 249_picture_grid.TXT
22923: 250_tipsandtricks2.txt
22924: 250_tipsandtricks3.txt
22925: 250_tipsandtricks3api.txt
22926: 250_tipsandtricks3_admin_elevation.txt
22927: 250_tipsandtricks3_tester.txt
22928: 250_tipsandtricks4_tester.txt
22929: 250_tipsandtricks4_tester2.txt
22930: 251_compare_noise_gauss.txt
22931: 251_whitenoise.txt
22932: 251_whitenoise2.txt
22933: 252_hilbert_turtle.txt
22934: 252_pas_hilbert.txt
22935: 253_opearatingsystem3.txt
22936: 254_dynarrays.txt
22937: 255_einstein.txt
22938: 256_findconsts_of_EXE.txt
22939: 256_findfunctions2_of_EXE.txt
22940: 256_findfunctions2_of_EXEaverp.txt
22941: 256_findfunctions2_of_EXEspec.txt
22942: 256_findfunctions3.txt
22943: 256_findfunctions_of_EXE.txt
22944: 257_AES_Cipher.txt
22945: 258_AES_cryptobox.txt
22946: 258_AES_cryptobox2.txt
22947: 258_AES_cryptobox2_passdlg.txt
22948: 259_AES_crypt_directory.txt
22949: 260_sendmessage_2.TXT
22950: 260_sendmessage_beta.TXT
22951: 261_probability.txt
22952: 262_mxoutputdemo4.txt
22953: 263_async_sound.txt
22954: 264_vclutils.txt
22955: 264_VCL_utils2.txt
22956: 265_timer_API.txt
22957: 266_serial_interface.txt
22958: 266_serial_interface2.txt
22959: 266_serial_interface3.txt
22960: 267_ackermann_rec.txt
22961: 267_ackermann_variants.txt
22962: 268_DBGrid_tree.txt
22963: 269_record_grid.TXT
22964: 270_Jedi_FunctionPower.txt
22965: 270_Jedi_FunctionPowertester.txt
22966: 271_closures_study.txt
22967: 271_closures_study_workingset2.txt
22968: 272_pas_function_show.txt
22969: 273_pas_function_show2.txt
22970: 274_library_functions.txt
22971: 275_turtle_language.txt
22972: 275_turtle_language_save.txt
22973: 276_save_algo.txt
22974: 276_save_algo2.txt
22975: 277_functionsfor39.txt
22976: 278_DB_Dialogs.TXT
22977: 279_hexer2.TXT
22978: 279_hexer2macro.TXT
22979: 279_hexer2macroback.TXT
22980: 280_UML_process.txt
22981: 280_UML_process_knabe2.txt
22982: 280_UML_process_knabe3.txt
22983: 280_UML_process_TIM_Botzenhardt.txt
22984: 280_UML_TIM_Seitz.txt
22985: 281_picturepuzzle.txt
22986: 281_picturepuzzle2.txt
22987: 281_picturepuzzle3.txt
22988: 281_picturepuzzle4.txt
22989: 479_inputquery.txt
22990: 480_regex_pathfinder2.txt
22991: 482 processPipe.txt
22992: 483_PathFuncTest_mX.txt
22993: 485_InnoFunc.txt
22994: 487_asyncKeyState.txt
22995: 489_simpleComport.txt
22996: 491_analogmeter.txt
22997: 493_gadgets.txt
22998: 496_InstallX.txt
22999: 498_UnitTesting.txt
23000: 500_diceoflifes.txt
23001: 502_findalldocs.txt
23002: 504_fileclass.txt
23003: 506_colormatrix.txt
23004: 508_simplecomportmorse.txt
23005: 509_509_GEOMap2_SReverse.TXT
23006: 511_LEDLabel.txt
23007: 513_StreamIntegration.txt

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

```
23008: 515_ledclock3.txt                              516_mapview.txt
23009: 517_animation7.txt                             518_sensors_meter.txt
23010: 519_powtils.txt                                520_run_bytecode.txt
23011:
23012:
23013: Web Script Examples:
23014:
23015: http://www.softwareschule.ch/examples/performer.txt';
23016: http://www.softwareschule.ch/examples/turtle.txt';
23017: http://www.softwareschule.ch/examples/SQLExport.txt';
23018: http://www.softwareschule.ch/examples/Richter.txt';
23019: http://www.softwareschule.ch/examples/checker.txt';
23020: http://www.softwareschule.ch/examples/demoscript.txt';
23021: http://www.softwareschule.ch/examples/ibzresult.txt';
23022: http://www.softwareschule.ch/examples/performindex.txt
23023: http://www.softwareschule.ch/examples/processlist.txt
23024: http://www.softwareschule.ch/examples/game.txt
23025:
23026:
23027:
23028: Delphi Basics Run Time Library listing
23029: ********************************************************************************
23030: A
23031: Compiler Directive $A Determines whether data is aligned or packed
23032: Compiler Directive $Align Determines whether data is aligned or packed
23033: Compiler Directive $AppType Determines the application type : GUI or Console
23034: Procedure SysUtils Abort Aborts the current processing with a silent exception
23035: Function System Abs Gives the absolute value of a number (-ve sign is removed)
23036: Directive Abstract Defines a class method only implemented in subclasses
23037: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
23038: Function System Addr Gives the address of a variable, function or procedure
23039: Keyword And Boolean and or bitwise and of two arguments
23040: Type System AnsiChar A character type guaranteed to be 8 bits in size
23041: Function SysUtils AnsiCompareStr Compare two strings for equality
23042: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
23043: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
23044: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
23045: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
23046: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
23047: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
23048: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
23049: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
23050: Function StrUtils AnsiPos Find the position of one string in another
23051: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
23052: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
23053: Function StrUtils AnsiRightStr Extracts characters from the right of a string
23054: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
23055: Type System AnsiString A data type that holds a string of AnsiChars
23056: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
23057: Procedure System Append Open a text file to allow appending of text to the end
23058: Procedure SysUtils AppendStr Concatenate one string onto the end of another
23059: Function Math ArcCos The Arc Cosine of a number, returned in radians
23060: Function Math ArcSin The Arc Sine of a number, returned in radians
23061: Function System ArcTan The Arc Tangent of a number, returned in radians
23062: Keyword Array A data type holding indexable collections of data
23063: Keyword As Used for casting object references
23064: Procedure System Assign Assigns a file handle to a binary or text file
23065: Function System Assigned Returns true if a reference is not nil
23066: Procedure System AssignFile Assigns a file handle to a binary or text file
23067: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
23068:
23069: B
23070: Compiler Directive $B Whether to short cut and and or operations
23071: Compiler Directive $BoolEval Whether to short cut and and or operations
23072: Procedure SysUtils Beep Make a beep sound
23073: Keyword Begin Keyword that starts a statement block
23074: Function System BeginThread Begins a separate thread of code execution
23075: Procedure System BlockRead Reads a block of data records from an untyped binary file
23076: Procedure System BlockWrite Writes a block of data records to an untyped binary file
23077: Type System Boolean Allows just True and False values
23078: Function Classes Bounds Create a TRect value from top left and size values
23079: Procedure System Break Forces a jump out of a single loop
23080: Type System Byte An integer type supporting values 0 to 255
23081:
23082: C
23083: Type System Cardinal The basic unsigned integer type
23084: Keyword Case A mechanism for acting upon different values of an Ordinal
23085: Function StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
23086: Function SysUtils ChangeFileExt Change the extension part of a file name
23087: Type System Char Variable type holding a single character
23088: Procedure System ChDir Change the working drive plus path for a specified drive
23089: Function System Chr Convert an integer into a character
23090: Keyword Class Starts the declaration of a type of object class
23091: Procedure System Close Closes an open file
23092: Procedure System CloseFile Closes an open file
23093: Variable System CmdLine Holds the execution text used to start the current program
23094: Type System Comp A 64 bit signed integer
23095: Function SysUtils CompareStr Compare two strings to see which is greater than the other
23096: Function SysUtils CompareText Compare two strings for equality, ignoring case
```

```
23097: Function Math CompareValue Compare numeric values with a tolerance
23098: Function System Concat Concatenates one or more strings into one string
23099: Keyword Const Starts the definition of fixed data values
23100: Keyword Constructor Defines the method used to create an object from a class
23101: Procedure System Continue Forces a jump to the next iteration of a loop
23102: Function ConvUtils Convert Convert one measurement value to another
23103: Function System Copy Create a copy of part of a string or an array
23104: Function System Cos The Cosine of a number
23105: Function SysUtils CreateDir Create a directory
23106: Type System Currency A floating point type with 4 decimals used for financial values
23107: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
23108: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
23109: Variable SysUtils CurrencyString The currency string used in currency display functions
23110: Function SysUtils CurrToStr Convert a currency value to a string
23111: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
23112:
23113: D
23114: Compiler Directive $D Determines whether application debug information is built
23115: Compiler Directive $DebugInfo Determines whether application debug information is built
23116: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
23117: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
23118: Function SysUtils Date Gives the current date
23119: Variable SysUtils DateSeparator The character used to separate display date fields
23120: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
23121: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
23122: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
23123: Function SysUtils DateToStr Converts a TDateTime date value to a string
23124: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
23125: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
23126: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
23127: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
23128: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
23129: Function DateUtils DaysInAMonth Gives the number of days in a month
23130: Function DateUtils DaysInAYear Gives the number of days in a year
23131: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
23132: Procedure System Dec Decrement an ordinal variable
23133: Variable SysUtils DecimalSeparator The character used to display the decimal point
23134: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
23135: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
23136: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
23137: Directive Default Defines default processing for a property
23138: Function Math DegToRad Convert a degrees value to radians
23139: Procedure System Delete Delete a section of characters from a string
23140: Function SysUtils DeleteFile Delete a file specified by its file name
23141: Keyword Destructor Defines the method used to destroy an object
23142: Function SysUtils DirectoryExists Returns true if the given directory exists
23143: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
23144: Function SysUtils DiskSize Gives the size in bytes of a specified drive
23145: Procedure System Dispose Dispose of storage used by a pointer type variable
23146: Keyword Div Performs integer division, discarding the remainder
23147: Keyword Do Defines the start of some controlled action
23148: Type System Double A floating point type supporting about 15 digits of precision
23149: Keyword DownTo Prefixes an decremental for loop target value
23150: Function StrUtils DupeString Creates a string containing copies of a substring
23151: Directive Dynamic Allows a class method to be overriden in derived classes
23152:
23153: E
23154: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
23155: Compiler Directive $EndIf Terminates conditional code compilation
23156: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
23157: Keyword Else Starts false section of if, case and try statements
23158: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
23159: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
23160: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
23161: Keyword End Keyword that terminates statement blocks
23162: Function DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
23163: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
23164: Procedure System EndThread Terminates a thread with an exit code
23165: Function System Eof Returns true if a file opened with Reset is at the end
23166: Function System Eoln Returns true if the current text file is pointing at a line end
23167: Procedure System Erase Erase a file
23168: Variable System ErrorAddr Sets the error address when an application terminates
23169: Keyword Except Starts the error trapping clause of a Try statement
23170: Procedure System Exclude Exclude a value in a set variable
23171: Procedure System Exit Exit abruptly from a function or procedure
23172: Variable System ExitCode Sets the return code when an application terminates
23173: Function System Exp Gives the exponent of a number
23174: Directive System Export Makes a function or procedure in a DLL externally available
23175: Type System Extended The floating point type with the highest capacity and precision
23176: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
23177: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
23178: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
23179: Function SysUtils ExtractFileName Extracts the name part of a full file name
23180: Function SysUtils ExtractFilePath Extracts the path part of a full file name
23181:
23182: F
23183: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
23184: Keyword File Defines a typed or untyped file
23185: Function SysUtils FileAge Get the last modified date/time of a file without opening it
```

```
23186: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
23187: Function SysUtils FileExists Returns true if the given file exists
23188: Function SysUtils FileGetAttr Gets the attributes of a file
23189: Variable System FileMode Defines how Reset opens a binary file
23190: Function System FilePos Gives the file position in a binary or text file
23191: Function SysUtils FileSearch Search for a file in one or more directories
23192: Function SysUtils FileSetAttr Sets the attributes of a file
23193: Function SysUtils FileSetDate Set the last modified date and time of a file
23194: Function System FileSize Gives the size in records of an open file
23195: Procedure System FillChar Fills out a section of storage with a fill character or byte value
23196: Keyword Finally Starts the unconditional code section of a Try statement
23197: Function SysUtils FindClose Closes a successful FindFirst file search
23198: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
23199: Function SysUtils FindFirst Finds all files matching a file mask and attributes
23200: Function SysUtils FindNext Find the next file after a successful FindFirst
23201: Function SysUtils FloatToStr Convert a floating point value to a string
23202: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
23203: Procedure System Flush Flushes buffered text file data to the file
23204: Keyword For Starts a loop that executes a finite number of times
23205: Function SysUtils ForceDirectories Create a new path of directories
23206: Function SysUtils Format Rich formatting of numbers and text into a string
23207: Function SysUtils FormatCurr Rich formatting of a currency value into a string
23208: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
23209: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
23210: Function System Frac The fractional part of a floating point number
23211: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
23212: Procedure System FreeMem Free memory storage used by a variable
23213: Keyword System Function Defines a subroutine that returns a value
23214:
23215: G
23216: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
23217: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
23218: Function System GetLastError Gives the error code of the last failing Windows API call
23219: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
23220: Function System GetMem Get a specified number of storage bytes
23221: Keyword Goto Forces a jump to a label, regardless of nesting
23222:
23223: H
23224: Compiler Directive $H Treat string types as AnsiString or ShortString
23225: Compiler Directive $Hints Determines whether Delphi shows compilation hints
23226: Procedure System Halt Terminates the program with an optional dialog
23227: Function System Hi Returns the hi-order byte of a (2 byte) Integer
23228: Function System High Returns the highest value of a type or variable
23229:
23230: I
23231: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
23232: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
23233: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
23234: Compiler Directive $IfOpt Tests for the state of a Compiler directive
23235: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
23236: Compiler Directive $IOChecks When on, an IO operation error throws an exception
23237: Keyword If Starts a conditional expression to determine what to do next
23238: Keyword Implementation Starts the implementation (code) section of a Unit
23239: Keyword In Used to test if a value is a member of a set
23240: Procedure System Inc Increment an ordinal variable
23241: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
23242: Procedure System Include Include a value in a set variable
23243: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
23244: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
23245: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
23246: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
23247: Function DateUtils IncYear Increments a TDateTime variable by a number of years
23248: Directive Index Principally defines indexed class data properties
23249: Constant Math Infinity Floating point value of infinite size
23250: Keyword Inherited Used to call the parent class constructor or destructor method
23251: Variable System Input Defines the standard input text file
23252: Function Dialogs InputBox Display a dialog that asks for user text input, with default
23253: Function Dialogs InputQuery Display a dialog that asks for user text input
23254: Procedure System Insert Insert a string into another string
23255: Function System Int The integer part of a floating point number as a float
23256: Type System Int64 A 64 bit sized integer - the largest in Delphi
23257: Type System Integer The basic Integer type
23258: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
23259: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
23260: Function SysUtils IntToStr Convert an integer into a string
23261: Function System IOResult Holds the return code of the last I/O operation
23262: Keyword Is Tests whether an object is a certain class or ascendant
23263: Function Math IsInfinite Checks whether a floating point number is infinite
23264: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
23265: Function System IsMultiThread Returns true if the code is running multiple threads
23266: Function Math IsNaN Checks to see if a floating point number holds a real number
23267:
23268: L
23269: Compiler Directive $L Determines what application debug information is built
23270: Compiler Directive $LocalSymbols Determines what application debug information is built
23271: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
23272: Function SysUtils LastDelimiter Find the last position of selected characters in a string
23273: Function System Length Return the number of elements in an array or string
23274: Function System Ln Gives the natural logarithm of a number
```

```
23275: Function System Lo Returns the low-order byte of a (2 byte) Integer
23276: Function Math Log10 Gives the log to base 10 of a number
23277: Variable SysUtils LongDateFormat Long version of the date to string format
23278: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
23279: Type System LongInt An Integer whose size is guaranteed to be 32 bits
23280: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
23281: Variable SysUtils LongTimeFormat Long version of the time to string format
23282: Type System LongWord A 32 bit unsigned integer
23283: Function System Low Returns the lowest value of a type or variable
23284: Function SysUtils LowerCase Change upper case characters in a string to lower case
23285:
23286: M
23287: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
23288: Function Math Max Gives the maximum of two integer values
23289: Constant System MaxInt The maximum value an Integer can have
23290: Constant System MaxLongInt The maximum value an LongInt can have
23291: Function Math Mean Gives the average for a set of numbers
23292: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
23293: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
23294: Function Math Min Gives the minimum of two integer values
23295: Constant SysUtils MinsPerDay Gives the number of minutes in a day
23296: Procedure System MkDir Make a directory
23297: Keyword Mod Performs integer division, returning the remainder
23298: Constant SysUtils MonthDays Gives the number of days in a month
23299: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
23300: Procedure System Move Copy bytes of data from a source to a destination
23301:
23302: N
23303: Constant Math NaN Not a real number
23304: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
23305: Procedure System New Create a new pointer type variable
23306: Constant System Nil A pointer value that is defined as undetermined
23307: Keyword Not Boolean Not or bitwise not of one arguments
23308: Function SysUtils Now Gives the current date and time
23309: Variable Variants Null A variable that has no value
23310:
23311: O
23312: Compiler Directive $O Determines whether Delphi optimises code when compiling
23313: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
23314: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
23315: Keyword System Object Allows a subroutine data type to refer to an object method
23316: Function System Odd Tests whether an integer has an odd value
23317: Keyword Of Linking keyword used in many places
23318: Keyword On Defines exception handling in a Try Except clause
23319: Keyword Or Boolean or or bitwise or of two arguments
23320: Function System Ord Provides the Ordinal value of an integer, character or enum
23321: Directive Out Identifies a routine parameter for output only
23322: Variable System Output Defines the standard output text file
23323: Directive Overload Allows 2 or more routines to have the same name
23324: Directive Override Defines a method that replaces a virtual parent class method
23325:
23326: P
23327: Keyword Packed Compacts complex data types into minimal storage
23328: Type System PAnsiChar A pointer to an AnsiChar value
23329: Type System PAnsiString Pointer to an AnsiString value
23330: Function System ParamCount Gives the number of parameters passed to the current program
23331: Function System ParamStr Returns one of the parameters used to run the current program
23332: Type System PChar A pointer to an Char value
23333: Type System PCurrency Pointer to a Currency value
23334: Type System PDateTime Pointer to a TDateTime value
23335: Type System PExtended Pointer to a Extended floating point value
23336: Function System Pi The mathematical constant
23337: Type System PInt64 Pointer to an Int64 value
23338: Function Classes Point Generates a TPoint value from X and Y values
23339: Type System Pointer Defines a general use Pointer to any memory based data
23340: Function Classes PointsEqual Compares two TPoint values for equality
23341: Function System Pos Find the position of one string in another
23342: Function System Pred Decrement an ordinal variable
23343: Function Printers Printer Returns a reference to the global Printer object
23344: Directive Private Starts the section of private data and methods in a class
23345: Keyword System Procedure Defines a subroutine that does not return a value
23346: Procedure FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
23347: Keyword System Program Defines the start of an application
23348: Function Dialogs PromptForFileName Shows a dialog allowing the user to select a file
23349: Keyword System Property Defines controlled access to class fields
23350: Directive Protected Starts a section of class private data accesible to sub-classes
23351: Type System PShortString A pointer to an ShortString value
23352: Type System PString Pointer to a String value
23353: Function Types PtInRect Tests to see if a point lies within a rectangle
23354: Directive Public Starts an externally accessible section of a class
23355: Directive Published Starts a published externally accessible section of a class
23356: Type System PVariant Pointer to a Variant value
23357: Type System PWideChar Pointer to a WideChar
23358: Type System PWideString Pointer to a WideString value
23359:
23360: Q
23361: Compiler Directive $Q Determines whether Delphi checks integer and enum bounds
23362:
23363: R
```

```
23364: Compiler Directive $R Determines whether Delphi checks array bounds
23365: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
23366: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
23367: Compiler Directive $Resource Defines a resource file to be included in the application linking
23368: Function Math RadToDeg Converts a radian value to degrees
23369: Keyword Raise Raise an exception
23370: Function System Random Generate a random floating point or integer number
23371: Procedure System Randomize Reposition the Random number generator next value
23372: Function Math RandomRange Generate a random integer number within a supplied range
23373: Variable System RandSeed Reposition the Random number generator next value
23374: Procedure System Read Read data from a binary or text file
23375: Procedure System ReadLn Read a complete line of data from a text file
23376: Type System Real A floating point type supporting about 15 digits of precision
23377: Type System Real48 The floating point type with the highest capacity and precision
23378: Procedure System ReallocMem Reallocate an existing block of storage
23379: Function DateUtils RecodeDate Change only the date part of a TDateTime variable
23380: Function DateUtils RecodeTime Change only the time part of a TDateTime variable
23381: Keyword Record A structured data type - holding fields of data
23382: Function Classes Rect Create a TRect value from 2 points or 4 coordinates
23383: Function SysUtils RemoveDir Remove a directory
23384: Procedure System Rename Rename a file
23385: Function SysUtils RenameFile Rename a file or directory
23386: Keyword Repeat Repeat statements until a ternmination condition is met
23387: Procedure SysUtils ReplaceDate Change only the date part of a TDateTime variable
23388: Procedure SysUtils ReplaceTime Change only the time part of a TDateTime variable
23389: Procedure System Reset Open a text file for reading, or binary file for read/write
23390: Variable System Result A variable used to hold the return value from a function
23391: Procedure System ReWrite Open a text or binary file for write access
23392: Procedure System RmDir Remove a directory
23393: Function System Round Rounds a floating point number to an integer
23394: Procedure System RunError Terminates the program with an error dialog
23395:
23396: S
23397: Constant SysUtils SecsPerDay Gives the number of seconds in a day
23398: Procedure System Seek Move the pointer in a binary file to a new record position
23399: Function System SeekEof Skip to the end of the current line or file
23400: Function System SeekEoln Skip to the end of the current line or file
23401: Function FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
23402: Variable System Self Hidden parameter to a method - refers to the containing object
23403: Keyword Set Defines a set of up to 255 distinct values
23404: Function SysUtils SetCurrentDir Change the current directory
23405: Procedure System SetLength Changes the size of a string, or the size(s) of an array
23406: Procedure System SetString Copies characters from a buffer into a string
23407: Keyword Shl Shift an integer value left by a number of bits
23408: Variable SysUtils ShortDateFormat Compact version of the date to string format
23409: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
23410: Type System ShortInt An integer type supporting values -128 to 127
23411: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
23412: Type System ShortString Defines a string of up to 255 characters
23413: Variable SysUtils ShortTimeFormat Short version of the time to string format
23414: Procedure Dialogs ShowMessage Display a string in a simple dialog with an OK button
23415: Procedure Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
23416: Procedure Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
23417: Keyword Shr Shift an integer value right by a number of bits
23418: Function System Sin The Sine of a number
23419: Type System Single The smallest capacity and precision floating point type
23420: Function System SizeOf Gives the storage byte size of a type or variable
23421: Function System Slice Creates a slice of an array as an Open Array parameter
23422: Type System SmallInt An Integer type supporting values from -32768 to 32767
23423: Function System Sqr Gives the square of a number
23424: Function System Sqrt Gives the square root of a number
23425: Procedure System Str Converts an integer or floating point number to a string
23426: Type System String A data type that holds a string of characters
23427: Function System StringOfChar Creates a string with one character repeated many times
23428: Function SysUtils StringReplace Replace one or more substrings found within a string
23429: Function System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
23430: Function SysUtils StrScan Searches for a specific character in a constant string
23431: Function SysUtils StrToCurr Convert a number string into a currency value
23432: Function SysUtils StrToDate Converts a date string into a TDateTime value
23433: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
23434: Function SysUtils StrToFloat Convert a number string into a floating point value
23435: Function SysUtils StrToInt Convert an integer string into an Integer value
23436: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
23437: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
23438: Function SysUtils StrToIntDef Convert a string into an Integer value with default
23439: Function SysUtils StrToTime Converts a time string into a TDateTime value
23440: Function StrUtils StuffString Replaces a part of one string with another
23441: Function System Succ Increment an ordinal variable
23442: Function Math Sum Return the sum of an array of floating point values
23443:
23444: T
23445: Function Math Tan The Tangent of a number
23446: Type Classes TBits An object that can hold an infinite number of Boolean values
23447: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
23448: Type ConvUtils TConvType Defines a measurement type as used by Convert
23449: Type System TDateTime Data type holding a date and time value
23450: Type System Text Defines a file as a text file
23451: Type System TextFile Declares a file type for storing lines of text
23452: Type SysUtils TFloatFormat Formats for use in floating point number display functions
```

```
23453: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
23454: Keyword Then Part of an if statement - starts the true clause
23455: Variable SysUtils ThousandSeparator The character used to display the thousands separator
23456: Keyword ThreadVar Defines variables that are given separate instances per thread
23457: Function SysUtils Time Gives the current time
23458: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
23459: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
23460: Variable SysUtils TimeSeparator The character used to separate display time fields
23461: Function SysUtils TimeToStr Converts a TDateTime time value to a string
23462: Type Classes TList General purpose container of a list of objects
23463: Keyword To Prefixes an incremental for loop target value
23464: Type System TObject The base class type that is ancestor to all other classes
23465: Function DateUtils Tomorrow Gives the date tomorrow
23466: Type Dialogs TOpenDialog Displays a file selection dialog
23467: Type Types TPoint Holds X and Y integer values
23468: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
23469: Type Types TRect Holds rectangle coordinate values
23470: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
23471: Function SysUtils Trim Removes leading and trailing blanks from a string
23472: Function SysUtils TrimLeft Removes leading blanks from a string
23473: Function SysUtils TrimRight Removes trailing blanks from a string
23474: Function System Trunc The integer part of a floating point number
23475: Procedure System Truncate Truncates a file size - removes all data after the current position
23476: Keyword Try Starts code that has error trapping
23477: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
23478: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
23479: Type Classes TStringList Holds a variable length list of strings
23480: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
23481: Type System TThreadFunc Defines the function to be called by BeginThread
23482: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
23483: Keyword Type Defines a new category of variable or process
23484:
23485:
23486: U
23487: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
23488: Keyword Unit Defines the start of a unit file - a Delphi module
23489: Keyword Until Ends a Repeat control loop
23490: Function System UpCase Convert a Char value to upper case
23491: Function SysUtils UpperCase Change lower case characters in a string to upper case
23492: Keyword Uses Declares a list of Units to be imported
23493:
23494: V
23495: Procedure System Val Converts number strings to integer and floating point values
23496: Keyword Var Starts the definition of a section of data variables
23497: Type System Variant A variable type that can hold changing data types
23498: Function Variants VarType Gives the current type of a Variant variable
23499: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
23500: Directive Virtual Allows a class method to be overriden in derived classes
23501:
23502: W
23503: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
23504: Keyword While Repeat statements whilst a continuation condition is met
23505: Type System WideChar Variable type holding a single International character
23506: Function System WideCharToString Copies a null terminated WideChar string to a normal string
23507: Type System WideString A data type that holds a string of WideChars
23508: Keyword With A means of simplifying references to structured variables
23509: Type System Word An integer type supporting values 0 to 65535
23510: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
23511: Procedure System Write Write data to a binary or text file
23512: Procedure System WriteLn Write a complete line of data to a text file
23513:
23514: X
23515: Compiler Directive $X Controls some Pascal extension handling
23516: Keyword Xor Boolean Xor or bitwise Xor of two arguments
23517:
23518: Y
23519: Compiler Directive $Y Determines whether application symbol information is built
23520: Function DateUtils Yesterday Gives the date yesterday
23521:
23522: Z
23523: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
23524:
23525: mapX:
23526:  if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
23527:     writeln('cologne map found');
23528:     GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
23529:     writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
23530:     OpenMapX('church trier');
23531:
23532: Signature:
23533: SHA1:  maXbox3.exe B1CFFA2139BEF2D27D17C8212331A1E8C7584BD2
23534: CRC32: maXbox3.exe 304FC6C2
23535: Ref:
23536:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
23537:     2. shdig: TSHA1Digest;
23538:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
23539:                for i:= 0 to 19 do write(BytetoHex(shdig[i]));
23540:
23541:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
```

```
23542:
23543: https://www.virustotal.com/en/file/...../
23544:
23545: SHA256:  17e0d19c9fadcc3c16cfcaac7b3e1453050dca8c83241d41f204da638ae4c113
23546: File name:  maXbox3.exe
23547: Compilation timestamp 2014-10-09 09:19:19
23548: Entry Point 0x01154AE0
23549: Number of sections 10
23550: Detection ratio:  2 / 55
23551: Analysis date:  2014-10-09 13:22:01 UTC
23552:
23553: PE sections
23554: Name Virtual address Virtual size Raw size Entropy MD5
23555: .text 4096 18117300 18117632 6.59 97869ce98a5942fc525a55c2636406a2
23556: .itext 18124800 48164 48640 6.56 8a260bdbd982c6a0c120da909cbabe77
23557: .data 18173952 262780 263168 5.57 14b2778a2160f34df1dbe4ebcd5a2473
23558: .bss 18440192 367600 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23559: .idata 18808832 54676 54784 5.52 b61cbd137919ed0e129d6a0b4450bf0f
23560: .edata 18866176 77 512 0.90 b9fb129a52e9590346006b22490f1aeb
23561: .tls 18870272 208 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23562: .rdata 18874368 24 512 0.26 2f53adef7a5532a1d844294ce179a52e
23563: .reloc 18878464 1153264 1153536 6.75 6f911e109eea583f7a15d5e84cfe1c12
23564: .rsrc 20033536 3430400 3430400 5.26 d90a71c9121d79482d68d6ba80f149d8
23565:
23566: File identification
23567: MD5 16b6cab86f976ebaf521afbe0a9f1ad0
23568: SHA1 b1cffa2139bef2d27d17c8212331a1e8c7584bd2
23569: SHA256 17e0d19c9fadcc3c16cfcaac7b3e1453050dca8c83241d41f204da638ae4c113
23570: ssdeep 393216:ORw/2P0kEm10mQreAepVwoQAqh4SPR4aJ/:OM2P0OnqoQ7hW
23571:
23572: authentihash dba614373e881ac632048dc60cf0e9199291ddce7afcb8c8c717ffd38cb2e94e
23573: imphash 5c32bd8a9083e008ac27ac5ea1657322
23574: File size 22.0 MB ( 23070208 bytes )
23575: File type Win32 EXE
23576: Magic literal
23577: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
23578:
23579: TrID  Windows ActiveX control (36.4%)
23580: Inno Setup installer (34.3%)
23581: InstallShield setup (13.4%)
23582: Win32 EXE PECompact compressed (generic) (13.0%)
23583: Win32 Executable (generic) (1.4%)
23584:
23585: Number of PE resources by type
23586: RT_BITMAP 699
23587: RT_STRING 304
23588: RT_RCDATA 154
23589: RT_ICON 49
23590: RT_GROUP_ICON 43
23591: RT_CURSOR 31
23592: RT_GROUP_CURSOR 26
23593: WAVE 9
23594: RT_DIALOG 2
23595: RT_HTML 2
23596: RT_MESSAGETABLE 1
23597: RT_MANIFEST 1
23598: RT_VERSION 1
23599:
23600:  ExifTool file metadata
23601: SpecialBuild
23602: mX4
23603: CodeSize 18166272
23604: SubsystemVersion 4.0
23605: Comments
23606: reduce to the max
23607: LinkerVersion 2.25
23608: ImageVersion 0.0
23609: FileSubtype 0
23610: FileVersionNumber 3.9.9.100
23611: LanguageCode
23612: German (Swiss)
23613: FileFlagsMask 0x003f
23614: FileDescription maXbox Delphi VM
23615: CharacterSet Windows, Latin1
23616: InitializedDataSize 4902912
23617: FileOS Win32
23618: TimeStamp 2014:10:09 10:19:19+01:00
23619: MIMEType application/octet-stream
23620:   ---- bigbitbox code_cleared_checked----
```