

```

1: ****
2: Constructor Function and Procedure List of maxbox 4.2.4.80 codeX signed
3: ****
4:
5: //////////////////////////////////////////////////////////////////
6: ref Help Extraxt of EXE Functions of maxbox4.exe BigBitBox API HEX in BOX4S
7: -----
8:
9: File EXE: 26.4 MB (27,720,144 bytes) V4.2.4.80 Oct 2016 to EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: *****Now the Funclist*****
11: Funclist Function : 17679 //15031 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 10532 //9968 //9172 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1686 //1492 //995 //
16: def head:max: maxBox10: 15/10/2016 10:04:41
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist424_.txt
18: doc file: maxbox_extract_funclist420.txt (sort function list)
19: -----
20: Funclist total Size all is: 29897! Constructor, Function and Procedure
21: AExtract of EXE Functions of maxbox4.exe, locs of file = 29615
22: ASize of EXE: 27,720,144 {26,650,112} {26,188,288} //25152000 (23614464) (22041600)
23: SHA1: of maXbox4.exe (4.2.4.80) 1556A557B0F9576AA5F23F2A1D06BE9699A757B
24:
25: https://www.virustotal.
   com/en/file/b80b0bfef22c6b4be3dbc4af984ca897144895f3c1e162f3ad7895d14fb4e667/analysis/1476479319/
26:
27: -----
28: -----
29: //////////////////////////////////////////////////////////////////
30:
31:
32: FUNCTION Metric of Script: 256 _findfunctions2_of_EXE.txt
33: Function *****Now the Funclist*****
34: function GetResStringChecked(Ident: string; const Args: array of const): string
35: Function ( Index : Longint ) : Integer
36: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
37: Function _CheckAutoResult( ResultCode : HResult ) : HResult
38: function _T(Name: tbtString): Variant;
39: function ABNFToText(const AText : String) : String
40: Function Abs(e : Extended) : Extended;
41: Function AbsInt(e : Extended) : Integer;
42: Function AbsInt( const B : integer ) : integer';
43: Function AbsFloat( const B : double ) : extended';
44: Function Ackermann( const A, B : Integer ) : Integer
45: Function AcquireLayoutLock : Boolean
46: Function ActionByName( const AName : string ) : TWebActionItem
47: Function ACTIVEBUFFER : PCHAR
48: Function Add : TAggregate
49: function Add : TCollectionItem
50: Function Add : TColumn
51: Function Add : TComboExItem
52: Function Add : TCookie
53: Function Add : TCoolBand
54: Function Add : TFavoriteLinkItem
55: Function Add : TFileTypeItem
56: Function Add : THeaderSection
57: Function Add : THTMLTableColumn
58: Function Add : TIdEMailAddressItem
59: Function Add : TIdMessagePart
60: Function Add : TIdUserAccount
61: Function Add : TListColumn
62: Function Add : TListItem
63: Function Add : TStatusPanel
64: Function Add : TTaskDialogBaseButtonItem
65: Function Add : TWebActionItem
66: Function Add : TWorkArea
67: Function Add( AClass : TClass ) : Integer
68: Function Add( AComponent : TComponent ) : Integer
69: Function Add( AItem, AData : Integer ) : Integer
70: Function Add( AItem, AData : Pointer ) : Pointer
71: Function Add( AItem, AData : TObject ) : TObject
72: Function Add( AObject : TObject ) : Integer
73: Function Add( const Access, Count : Cardinal; const Offset : Int64 ) : Integer
74: Function Add( const S : WideString ) : Integer
75: Function Add( Image, Mask : TBitmap ) : Integer
76: Function Add( Index : LongInt; const Text : string ) : LongInt
77: Function Add( Sibling : TTreenode; const S : string ) : TTreenode
78: Function Add( const S : string ) : Integer
79: function Add(S: string): Integer;
80: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer ) : Integer
81: Function ADDCHILD : TFIELDDEF
82: Function AddChild( Index : LongInt; const Text : string ) : LongInt
83: Function AddChild( Parent : TTreenode; const S : string ) : TTreenode
84: Function AddChildObject( Parent : TTreenode; const S : string ) : TTreenode
85: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer ) : LongInt
86: Function AddChildObject( Parent : TTreenode; const S : string; Ptr : Pointer ) : TTreenode
87: Function AddChildObjectFirst( Parent : TTreenode; const S : string; Ptr : Pointer ) : TTreenode
88: Function ADDFIELDDEF : TFIELDDEF

```

```

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

```

```

177: Function AnsiStrScan(Str: PChar; Chr: Char): PChar
178: Function AnsiStrUpper( Str : PChar) : PChar
179: Function AnsiToUtf8( const S : string) : UTF8String
180: Function AnsiToUtf8Ex( const S : string; const cp : integer) : UTF8String
181: Function AnsiUpperCase( S : string) : string
182: Function AnsiUpperCaseFile( S : string) : string
183: Function ApplyUpdates(const Delta: OleVariant;MaxErrors:Integer; out ErrorCount: Integer): OleVariant
185: Function ApplyUpdates(const Delta:OleVariant;MaxErrors: Integer;out ErrorCount: Integer) : OleVariant;
186: Function ApplyUpdates( MaxErrors : Integer ) : Integer
187: Function ApplyUpdates1(const Delta:OleVar;MaxErrs:Int;out ErrCount:Int,var OwnerData:OleVar):OleVariant;
188: Function ArcCos( const X : Extended) : Extended
189: Function ArcCot( const X : Extended) : Extended
190: Function ArcCotH( const X : Extended) : Extended
192: Function ArcCsc( const X : Extended) : Extended
193: Function ArcCscH( const X : Extended) : Extended
194: Function ArcSec( const X : Extended) : Extended
195: Function ArcSecH( const X : Extended) : Extended
196: Function ArcSin( const X : Extended) : Extended
197: Function ArcSinh( const X : Extended) : Extended
198: Function ArcTan( const X : Extended) : Extended
199: Function ArcTan2( const Y, X : Extended) : Extended
200: Function ArithmeticMean( const X : TDynDoubleArray) : Float
201: function ArrayLength: integer;
202: Function AsHex( const AValue : T4x4LongWordRecord) : string
203: Function AsHex( const AValue : T5x4LongWordRecord) : string
204: Function ASNDecLen( var Start : Integer; const Buffer : string) : Integer
205: Function ASNDecOIDItem( var Start : Integer; const Buffer : string) : Integer
206: Function ASNEncInt( Value : Integer) : string
207: Function ASNEncLen( Len : Integer) : string
208: Function ASNEncOIDItem( Value : Integer) : string
209: Function ASNEncUInt( Value : Integer) : string
210: Function ASNItem( var Start : Integer; const Buffer : string; var ValueType : Integer) : string
211: Function ASNObject( const Data : string; ASNType : Integer) : string
212: Function Assigned(I: Longint): Boolean;
213: Function AspectRatio(aWidth, aHeight: Integer): String;
214: Function AsWideString( Field : TField) : WideString
215: Function AtLeast( ACount : Integer) : Boolean
216: Function AttemptToUseSharedMemoryManager : Boolean
217: Function Authenticate : Boolean
218: Function AuthenticateUser( const AUsername, APassword : String) : Boolean
219: Function Authentication : String
220: Function BatchMove( ASource : TBDEDataSet; AMode : TBatchMode) : Longint
221: Function BcdCompare( const bcd1, bcd2 : TBcd) : Integer
222: Function BcdFromBytes( const AValue : TBytes) : TBcd
223: Function BcdPrecision( const Bcd : TBcd) : Word
224: Function BcdScale( const Bcd : TBcd) : Word
225: Function BcdToBytes( const Value : TBcd) : TBytes
226: Function BCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
227: Function BcdToDouble( const Bcd : TBcd) : Double
228: Function BcdToInteger( const Bcd : TBcd; Truncate : Boolean) : Integer
229: Function BcdToStr( const Bcd : TBcd) : string;
230: Function BcdToStrF(const Bcd : TBcd; Format: TFloatFormat; const Precision, Digits:Integer):string
231: function beep2(dwFreq, dwDuration: integer): boolean;
232: Function BeginPeriod( const Period : Cardinal) : Boolean
233: Function BeginTrans : Integer
234: Function BeginTransaction : TDBXTransaction;
235: Function BeginTransaction1( Isolation : TDBXIsolation) : TDBXTransaction;
236: function BigMulu(aone, atwo: string): string;
237: function BigNumber(aone, atwo: string): string;
238: function BigExp(aone, atwo: string): string;
239: function BigMul(aone, atwo: string): string;
240: function BigAdd(aone, atwo: string): string;
241: function BigSub(aone, atwo: string): string;
242: function BigFactorial(aone: string): string;
243: Function Binary.ToDouble( ABinary : string; DefValue : Double) : Double
244: Function BinomialCoeff( N, R : Cardinal) : Float
245: function BinomialCoefficient(n, k: Integer): string;
246: Function BinStrToInt( const ABinary : String) : Integer
247: Function BinToByte(Binary: String): Byte;
248: function BinToHex2(Binary: String): string;
249: function BinToInt(Binary: String): Integer;
250: Function BinToChar(St: String): Char;
251: Function BinToStr(ans: string): string;
252: Function BitBlt(hdcDest:HDC;nXDest,nYDest,nWidth,nHeigh:Int;hdcSrc:HDC;nXSrc,nYSrc:Int;dwRop:DWORD):Bool;
253: Function BitmapsAreIdentical( ABitmap1, ABitmap2 : TBitmap) : Boolean
254: Function BitsHighest( X : Byte) : Integer;
255: Function BitsHighest1( X : ShortInt) : Integer;
256: Function BitsHighest2( X : SmallInt) : Integer;
257: Function BitsHighest3( X : Word) : Integer;
258: Function BitsHighest4( X : Integer) : Integer;
259: Function BitsHighest5( X : Cardinal) : Integer;
260: Function BitsHighest6( X : Int64) : Integer;
261: Function BitsLowest( X : Byte) : Integer;
262: Function BitsLowest1( X : Shortint) : Integer;
263: Function BitsLowest2( X : Smallint) : Integer;
264: Function BitsLowest3( X : Word) : Integer;
265: Function BitsLowest4( X : Cardinal) : Integer;

```

```

266: Function BitsLowest5( X : Integer) : Integer;
267: Function BitsLowest6( X : Int64) : Integer;
268: Function BitsNeeded( const X : Byte) : Integer;
269: Function BitsNeeded1( const X : Word) : Integer;
270: Function BitsNeeded2( const X : Integer) : Integer;
271: Function BitsNeeded3( const X : Int64) : Integer;
272: Function BlueComponent( const Color32 : TColor32) : Integer;
273: Function BooleanToInteger( const Pb : Boolean) : Integer;
274: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string;
275: Function BoolToStr1(value : boolean) : string;
276: Function booltoint( aBool : Boolean) : LongInt;
277: Function inttobool( aInt : LongInt) : Boolean;
278: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect;
279: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect;
280: Function BreakApart( BaseString, BreakString : string; StringList : TStrings) : TStrings;
281: Function BrightColor( const Color : TColor; const Pct : Single) : TColor;
282: Function BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte;
283: Function BufferRequest( Length : Integer) : TStream;
284: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean;
285: Function Buttons : PTaskDialogButton;
286: Function BytesPerScanline( PixelsPerScanline, BitsPerPixel, Alignment : Longint) : Longint;
287: Function BytesToCardinal( const AValue : TIddBytes; const AIndex : Integer) : Cardinal;
288: Function BytesToChar( const AValue : TIddBytes; const AIndex : Integer) : Char;
289: Function BytesToInt64( const AValue : TIddBytes; const AIndex : Integer) : Int64;
290: Function BytesToInteger( const AValue : TIddBytes; const AIndex : Integer) : Integer;
291: Function BytesToIPv6( const AValue : TIddBytes; const AIndex : Integer) : TIIdIpv6Address;
292: Function BytesToShort( const AValue : TIddBytes; const AIndex : Integer) : Short;
293: Function BytesToString(ABytes:TIddBytes; AStartIndex:Integer; AMaxCount:Integer): string;
294: Function BytesToStr(const Value: TBytes): String;
295: Function BytesToWord( const AValue : TIddBytes; const AIndex : Integer) : Word;
296: Function ByteToBin(Int: Byte): String;
297: Function ByteToCharIndex( S : string; Index : Integer) : Integer;
298: function ByteToCharIndex(const S: string; Index: Integer): Integer;
299: Function ByteToCharLen( S : string; MaxLen : Integer) : Integer;
300: function ByteToCharLen(const S: string; MaxLen: Integer): Integer;
301: Function ByteToHex( const AByte : Byte) : string;
302: Function ByteToOctal( const AByte : Byte) : string;
303: Function ByteType( S : string; Index : Integer) : TMbcsByteType;
304: function ByteType(const S: string; Index: Integer): TMbcsByteType;
305: Function CalcTitleRect( Col : TColumn; ARow : Integer; var MasterCol : TColumn) : TRect;
306: Function CalculateDFAFingerprint( oStates : TList) : integer;
307: function CallTerminateProcs: Boolean;
308: function CANFOCUS:BOOLEAN;
309: Function CanLoad( const Ext : string) : Boolean;
310: Function CanParse( AWebRequest : TWebRequest) : Boolean;
311: Function CanSave( const Ext : string) : Boolean;
312: Function CanStart( cChar : char) : boolean;
313: Function CaptureScreen : TBitmap;
314: Function CaptureScreen1( Rec : TRect) : TBitmap;
315: Function CardinalToFourChar( ACardinal : LongWord) : string;
316: Function CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Boolean): Boolean;
317: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
318: Function Ceil( const X : Extended) : Integer;
319: Function Ceil16( X : Integer) : Integer;
320: Function Ceil4( X : Integer) : Integer;
321: Function Ceil8( X : Integer) : Integer;
322: Function Ceiling( const X : Extended) : Integer;
323: Function CellRect( ACol, ARow : Longint) : TRect;
324: Function CelsiusToFahrenheit( const AValue : Double) : Double;
325: Function CenterPoint( const Rect : TRect) : TPoint;
326: function CenterPoint(const Rect: TRect): TPoint;
327: Function ChangeFileExt( FileName, Extension : string) : string;
328: function ChangeFileExt(const FileName: string; const Extension: string): string;
329: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean;
330: Function CharInSet( const Ch : Char; const testSet: TSysCharSet): Boolean;
331: Function CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean;
332: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean;
333: Function CharLength( S : String; Index : Integer) : Integer;
334: Function CharRange( const AMin, AMax : Char) : String;
335: function CharsetToIdent(Charset: Longint; var Ident: string): Boolean;
336: Function CharToBin(vChr: Char): String;
337: Function CharNext(lpsz: PChar): PChar; stdcall;
338: Function CharToByteIndex( S : string; Index : Integer) : Integer;
339: function CharToByteIndex(const S: string; Index: Integer): Integer;
340: Function CharToByteLen( S : string; MaxLen : Integer) : Integer;
341: function CharToByteLen(const S: string; MaxLen: Integer): Integer;
342: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
343: function CharToHexStr(Value: char): string;
344: function CharToOem(ins, outs: PChar):boolean;
345: function CharToUniCode(Value: Char): string;
346: Function CheckMenuDropdown : Boolean;
347: Function CheckMessages : longint;
348: Function CheckBox: string;
349: Function CheckOpen( Status : DBIResult) : Boolean;
350: Function CheckPassword( const APassword : String) : Boolean;
351: Function CheckResponse( const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt;
352: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
353: function CheckSynchronize(Timeout: Integer): Boolean;
354: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean;

```

```

355: Function CheckCom(AComNumber: Integer): Integer;')
356: Function CheckLPT1: string;');
357: function ChrA(const a: byte): char;
358: Function ClassIDToProgID(const ClassID: TGUID): string;
359: Function ClassNameIs(const Name: string): Boolean;
360: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
361: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
362: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
363: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
364: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
365: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
366: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
367: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
368: Function Clipboard : TClipboard
369: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
370: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
371: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
372: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
373: Function Clone( out stm : IStream) : HResult
374: Function CloneConnection : TSQLConnection
375: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
376: function CLOSEQUERY:BOOLEAN
377: Function CloseVolume( var Volume : THandle) : Boolean
378: Function CloseHandle(Handle: Integer): Integer; stdcall;
379: Function CPLApplet( hwndCPL : THandle; uMsg : DWORD; lParam1, lParam2 : Longint) : Longint
380: Function CmdLine: PChar;
381: function CmdShow: Integer;
382: // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
383: Function CoordinateStr(Idx: Integer; PosInSec: Double; PosIn: TPos): string;
384: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
385: Function Color32( WinColor : TColor) : TColor32;
386: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
387: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale : Boolean) : TColor;
388: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef;
389: Function ColorToHTML( const Color : TColor) : String;
390: function ColorToIdent(Color: Longint; var Ident: string): Boolean;
391: Function ColorToRGB(color: TColor): Longint;
392: function ColorToString(Color: TColor): string;
393: Function ColorToWebColorName( Color : TColor) : string;
394: Function ColorToWebColorStr( Color : TColor) : string;
395: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn;
396: Function Combination(npr, ncr: integer): extended;
397: Function CombinationInt(npr, ncr: integer): Int64;
398: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo;
399: Function CommaAdd( const AStr1, AStr2 : String) : string;
400: Function CommercialRound( const X : Extended) : Int64;
401: Function Commit( grfCommitFlags : Longint) : HResult;
402: Function Compare( const NameExt : string) : Boolean;
403: function CompareDate(const A, B: TDateTime): TValueRelationship;
404: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer;
405: Function CompareFiles(const FN1,FN2:string;Breathe:TNotifyEvent;BreathingSender:TObject): boolean;
406: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean;
407: Function CompareStr( S1, S2 : string) : Integer;
408: function CompareStr(const S1: string; const S2: string): Integer;
409: function CompareString(const S1: string; const S2: string): Integer;
410: Function CompareText( S1, S2 : string) : Integer;
411: function CompareText( const S1: string; const S2: string): Integer;
412: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean;
413: function CompareTime( const A, B: TDateTime): TValueRelationship;
414: function CompareValueE(const A, B:Extended; Epsilon: Extended = 0): TValueRelationship; overload;
415: function CompareValueD(const A, B: Double; Epsilon: Double = 0): TValueRelationship; overload;
416: function CompareValueS(const A, B: Single; Epsilon: Single = 0): TValueRelationship; overload;
417: function CompareValueI(const A, B: Integer): TValueRelationship; overload;
418: function CompareValueI64(const A, B: Int64): TValueRelationship; overload;
419: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean;
420: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean;
421: Function ComponentTypeToString( const ComponentType : DWORD) : string;
422: Function ComposeDateTime(Date,Time : TDateTime) : TDateTime;';
423: Function ComponentToStringProc(Component: TComponent): string;
424: Function StringToComponentProc(Value: string): TComponent;
425: Function CompToCurrency( Value : Comp) : Currency;
426: Function Comp.ToDouble( Value : Comp) : Double;
427: function ComputeFileCRC32(const FileName : String) : Integer;
428: function ComputeSHA256(astr: string; amode: char): string) //mode F:File, S:String;
429: function ComputeSHA512(astr: string; amode: char): string) //mode F:File, S:String;
430: function ComPortSelect: Integer; // Search for the first available port;
431: Function Concat(s: string): string;
432: Function ConnectAndGetAll : string;
433: Function Connected : Boolean;
434: function constrain(x, a, b: integer): integer;
435: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult;
436: Function ConstraintsDisabled : Boolean;
437: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN;
438: Function ContainsState( oState : TniRegularExpressionState) : boolean;
439: Function ContainsStr( const AText, ASubText : string) : Boolean;
440: Function ContainsText( const AText, ASubText : string) : Boolean;
441: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean;
442: Function Content : string;
443: Function ContentFromStream( Stream : TStream) : string;

```

```

444: Function ContentFromString( const S : string) : string
445: Function CONTROLSDISABLED : BOOLEAN
446: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
447: Function ConvertI( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
448: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
449: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
450: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
451: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
452: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
453: Function ConvTypeToDescription( const AType : TConvType) : string
454: Function ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
455: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
456: Function ConvAdd(const AVal:Dbl;const AType1:TConvType;const AVal2:Dbl;const AType2,
AResultType:TConvType): Double
457: Function ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
AType2:TConvType): TValueRelationship
458: Function ConvDec( const AValue : Double; const AType, AAmountType : TConvType) : Double;
459: Function ConvDecl(const AValue:Dbl;const AType:TConvType;const AAmt:Dble;const
AAmountType:TConvType):Double;
460: Function ConvDiff(const AVal1:Dbl;const AType1:TConvType;const AVal2:Dble;const AType2,
AResType:TConvType):Double
461: Function ConvInc( const AValue : Double; const AType, AAmtType : TConvType) : Double;
462: Function ConvIncl(const AValue:Dbl;const AType:TConvType;const AAmt:Double;const
AAmtType:TConvType):Double;
463: Function ConvSameValue(const AValue1:Dbl;const AType1:TConvType;const AValue2:Dbl;const
AType2:TConvType):Bool
464: Function ConvToStr( const AValue : Double; const AType : TConvType) : string
465: Function ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmt : Double;
const AAmtType : TConvType) : Boolean
466: Function ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType; const AAmt:Double;const
AAmtType: TConvType) : Boolean
467: function Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
468: Function CopyFile( Source, Dest : string; CanOverwrite : Boolean) : Boolean
469: Function CopyFileEx( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
470: Function CopyfileTo( const Source, Destination : string) : Boolean
471: function CopyFrom(Source:TStream;Count:Int64):LongInt
472: Function CopyPalette( Palette : HPALETTE) : HPALETTE
473: Function CopyTo( Length : Integer) : string
474: Function CopyTo(stm: IStream; cb: Largeint;out cbRead: Largeint;out cbWritten:Largeint): HResult
475: Function CopyToEOF : string
476: Function CopyToEOL : string
477: Function Cos(e : Extended) : Extended;
478: Function Cosecant( const X : Extended) : Extended
479: Function Cot( const X : Extended) : Extended
480: Function Cotan( const X : Extended) : Extended
481: Function CotH( const X : Extended) : Extended
482: Function Count : Integer
483: Function CountBitsCleared( X : Byte) : Integer;
484: Function CountBitsCleared1( X : Shortint) : Integer;
485: Function CountBitsCleared2( X : Smallint) : Integer;
486: Function CountBitsCleared3( X : Word) : Integer;
487: Function CountBitsCleared4( X : Integer) : Integer;
488: Function CountBitsCleared5( X : Cardinal) : Integer;
489: Function CountBitsCleared6( X : Int64) : Integer;
490: Function CountBitsSet( X : Byte) : Integer;
491: Function CountBitsSet1( X : Word) : Integer;
492: Function CountBitsSet2( X : Smallint) : Integer;
493: Function CountBitsSet3( X : ShortInt) : Integer;
494: Function CountBitsSet4( X : Integer) : Integer;
495: Function CountBitsSet5( X : Cardinal) : Integer;
496: Function CountBitsSet6( X : Int64) : Integer;
497: function countDirfiles(const apath: string): integer;
498: function CountGenerations(Ancestor,Descendent: TClass): Integer
499: Function Coversine( X : Float) : Float
500: function CRC32(const fileName: string): LongWord;
501: Function CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
502: Function CreateColumns : TDBGridColumns
503: Function CreateDataLink : TGridDataLink
504: Function CreateDir( Dir : string) : Boolean
505: function CreateDir(const Dir: string): Boolean
506: Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
507: Function CreateEnvironmentBlock(const Options:TEnvironmentOptions;const AdditionalVars:TStrings): PChar
508: Function CREATEFIELD(OWNER:TCOMPONENT;PARENTFIELD:TOBJECTFIELD;const
FIELDNAME:String;CREATECHILDREN:BOOL):TFIELD
509: Function CreateGlobber( sFilespec : string) : TniRegularExpression
510: Function CreateGrayMappedBmp( Handle : HBITMAP) : HBITMAP
511: Function CreateGrayMappedRes( Instance : THandle; ResName : PChar) : HBITMAP
512: function CreateGUID(out Guid: TGUID): HResult
513: Function CreateInstance( const unkOuter : IUnknown; const iid : TGUID; out obj) : HResult
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 CreateMessageDialog(const Msg:string; DlgType:TMsgDlgType; Buttons: TMsgDlgButtons) : TForm;
517: Function CreateMessageDialog1(const
Msg:string;DlgType:TMsgDlgType;Btns:TMsgDlgBtns;DefaultBtn:TMsgDlgBtn):TForm;
518: function CreateOleObject(const ClassName: String): IDispatch;
519: Function CREATEPARAM( FLDTYPE : TFIELDTYPE; const PARAMNAME : String; PARAMTYPE : TPARAMTYPE) : TPARAM
520: Function CreateParameter(const
Name:WideString;DataType:TDataType;Direction:TParameterDirection;Size:Integer;Value: OleVariant):TParameter
521: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject

```

```

522: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
523: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
524: Function CreateMutex2(lpMutexAttributes:TObject;bInitialOwner: BOOL;lpName : PChar) : THandle';
525:   Function CreateSemaphore2( lpSemaphoreAttributes: TObject; lInitialCount, lMaximumCount : Longint;
      lpName : PChar) : THandle';
526:   //Ex.: vMutex := CreateMutex2(Nil, True, MutexName);
527: Function IPAddrToName // ShowMessage(IPAddrToName(LocalIp));
528: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
529: Function CreateValueBuffer( Length : Integer) : TValueBuffer
530: Function CreatePopupCalculator( Aowner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
531: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
532: Function CreateRotatedFont( Font : TFont; Angle : Integer) : HFONT
533: Function CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
534: Function CreateValueBuffer( Length : Integer) : TValueBuffer
535: Function CreateHexDump( Aowner : TWinControl) : THexDump
536: Function Csc( const X : Extended) : Extended
537: Function CscH( const X : Extended) : Extended
538: function currencyDecimals: Byte
539: function currencyFormat: Byte
540: function currencyString: String
541: Function CurrentProcessId : TIdPID
542: Function CurrentReadBuffer : string
543: Function CurrentThreadId : TIdPID
544: Function CurrentYear : Word
545: Function CurrToBCD(const Curr: Currency; var BCD: TBcd; Precision: Integer; Decimals: Integer): Boolean
546: Function CurrToStr( Value : Currency) : string;
547: Function CurrToStrF( Value : Currency; FormatSettings : TFormatSettings; Digits: Integer) : string;
548: Function CurrToStrFS(Value:Currency;Format:TFloatFormat;Digits:Int;const
  FormatSettings:TFormatSettings):string;
549: function CursorToIdent(cursor: Longint; var Ident: string): Boolean;
550: function CursorToString(cursor: TCursor): string;
551: Function CustomSort( SortProc : TLVCompare; lParam : Longint) : Boolean
552: Function CustomSort( SortProc : TTVCompare; Data : Longint; ARecurse : Boolean) : Boolean
553: Function CycleToDeg( const Cycles : Extended) : Extended
554: Function CycleToGrad( const Cycles : Extended) : Extended
555: Function CycleToRad( const Cycles : Extended) : Extended
556: Function D2H( N : Longint; A : Byte) : string
557: Function DarkColor( const Color : TColor; const Pct : Single) : TColor
558: Function DarkColorChannel( const Channel : Byte; const Pct : Single) : Byte
559: Function DatalinkDir : string
560: Function DataRequest( Data : OleVariant) : OleVariant
561: Function DataRequest( Input : OleVariant) : OleVariant
562: Function DataToRawColumn( ACol : Integer) : Integer
563: Function Date : TDateTime
564: function Date: TDateTime;
565: Function DateIsNull( const pdtValue : TDateTime; const pdtKind : TdtKind) : Boolean
566: Function DateOf( const AValue : TDateTime) : TDateTime
567: function DateSeparator: char;
568: Function DateTimeGMTToHttpStr( const GMTValue : TDateTime) : String
569: Function DatetimeToFileDate( DateTime : TDateTime) : Integer
570: function DatetimeToFileDate(DateTime: TDateTime): Integer;
571: Function DateTimeToGmtOffsetStr( ADateTime : TDateTime; SubGMT : Boolean) : string
572: Function DateTimeToInternetStr( const Value : TDateTime; const AIsgmt : Boolean) : String
573: Function DatetimeToJulianDate( const AValue : TDateTime) : Double
574: Function DateTimeToModifiedJulianDate( const AValue : TDateTime) : Double
575: Function DateTimeToStr( Datetime : TDateTime) : string;
576: Function DateTimeToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings) : string;
577: function DatetimeToTimeStamp(DateTime: TDateTime): TTimeStamp
578: Function DateTimeToUnix( const AValue : TDateTime) : Int64
579: function DateTimeToUnix(D: TDateTime): Int64;
580: Function DateToStr( DateTime : TDateTime) : string;
581: function DateToStr(const DateTime: TDateTime): string;
582: function DateToStr(D: TDateTime): string;
583: Function DateToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings) : string;
584: Function DayOf( const AValue : TDateTime) : Word
585: Function DayOfTheMonth( const AValue : TDateTime) : Word
586: function DayOfTheMonth(const AValue: TDateTime): Word;
587: Function DayOfTheWeek( const AValue : TDateTime) : Word
588: Function DayOfTheYear( const AValue : TDateTime) : Word
589: function DayOfTheYear(const AValue: TDateTime): Word;
590: Function DayOfWeek( DateTime : TDateTime) : Word
591: function DayOfWeek(const DateTime: TDateTime): Word;
592: Function DayOfWeekStr( DateTime : TDateTime) : string
593: Function DaysBetween( const ANow, AThen : TDateTime) : Integer
594: Function DaysInAMonth( const AYear, AMonth : Word) : Word
595: Function DaysInAYear( const AYear : Word) : Word
596: Function DaysInMonth( const AValue : TDateTime) : Word
597: Function DaysInYear( const AValue : TDateTime) : Word
598: Function DaySpan( const ANow, AThen : TDateTime) : Double
599: Function DBUseRightToLeftAlignment( AControl : TControl; AField : TField) : Boolean
600: function DecimalSeparator: char;
601: Function DecLimit( var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;
602: Function DecLimit1(var B : Shortint; const Limit : Shortint; const Decr : Shortint) : Shortint;
603: Function DecLimit2(var B : Smallint; const Limit : Smallint; const Decr : Smallint) : Smallint;
604: Function DecLimit3( var B : Word; const Limit : Word; const Decr : Word) : Word;
605: Function DecLimit4( var B : Integer; const Limit : Integer; const Decr : Integer) : Integer;
606: Function DecLimit5(var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal) : Cardinal;
607: Function DecLimit6( var B : Int64; const Limit : Int64; const Decr : Int64) : Int64;
608: Function DecLimitClamp( var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;

```

```

609: Function DecLimitClamp1( var B : Shortint; const Limit : Shortint; const Decr : Shortint) : Shortint;
610: Function DecLimitClamp2( var B : Smallint; const Limit : Smallint; const Decr : Smallint) : Smallint;
611: Function DecLimitClamp3( var B : Word; const Limit : Word; const Decr : Word) : Word;
612: Function DecLimitClamp4( var B : Integer; const Limit : Integer; const Decr : Integer) : Integer;
613: Function DecLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal) : Cardinal;
614: Function DecLimitClamp6( var B : Int64; const Limit : Int64; const Decr : Int64) : Int64;
615: Function DecodeDateFully( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
616: Function DecodeSoundexInt( AValue : Integer) : string
617: Function DecodeSoundexWord( AValue : Word) : string
618: Function DefaultAlignment : TAlignment
619: Function DefaultCaption : string
620: Function DefaultColor : TColor
621: Function DefaultFont : TFont
622: Function DefaultIMEMode : TIMEMode
623: Function DefaultIMEName : TIMEName
624: Function DefaultReadOnly : Boolean
625: Function DefaultWidth : Integer
626: Function DegMinSecToFloat( const Degs, Mins, Secs : Float) : Float
627: Function DegToCycle( const Degrees : Extended) : Extended
628: Function DegToGrad( const Degrees : Extended) : Extended
629: Function DegToGrad( const Value : Extended) : Extended;
630: Function DegToGrad1( const Value : Double) : Double;
631: Function DegToGrad2( const Value : Single) : Single;
632: Function DegToRad( const Degrees : Extended) : Extended
633: Function DegToRad( const Value : Extended) : Extended;
634: Function DegToRad1( const Value : Double) : Double;
635: Function DegToRad2( const Value : Single) : Single;
636: Function DelChar( const pStr : string; const pChar : Char) : string
637: Function DelEnvironmentVar( const Name : string) : Boolean
638: Function Delete( const MsgNum : Integer) : Boolean
639: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
640: Function DeleteFile(const FileName: string): boolean
641: Function DeleteFileEx(FileName: string; Flags: FILEOP_FLAGS) : Boolean
642: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
643: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char): integer;
644: Function DelSpace( const pStr : string) : string
645: Function DelString( const pStr, pDelStr : string) : string
646: Function DelTree( const Path : string) : Boolean
647: Function Depth : Integer
648: Function Description : string
649: Function DescriptionsAvailable : Boolean
650: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
651: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
652: Function DescriptionToConvTypeL(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Bool;
653: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
654: Function DialogsToPixelsX( const Dialogs : Word) : Word
655: Function DialogsToPixelsY( const Dialogs : Word) : Word
656: Function Digits( const X : Cardinal) : Integer
657: Function DirectoryExists( const Name : string) : Boolean
658: Function DirectoryExists( Directory : string) : Boolean
659: Function DiskFree( Drive : Byte) : Int64
660: function DiskFree(Drive: Byte): Int64
661: Function DiskInDrive( Drive : Char) : Boolean
662: Function DiskSize( Drive : Byte) : Int64
663: function DiskSize(Drive: Byte): Int64)
664: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
665: Function DispatchEnabled : Boolean
666: Function DispatchMask : TMask
667: Function DispatchMethodType : TMethodType
668: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
669: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
670: Function DisplayCase( const S : String) : String
671: Function DisplayRect( Code : TDisplayCode) : TRect
672: Function DisplayRect( TextOnly : Boolean) : TRect
673: Function DisplayStream( Stream : TStream) : string
674: TBufferCoord', 'record Char : integer; Line : integer; end
675: TDisplayCoord', 'record Column : integer; Row : integer; end
676: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord
677: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord
678: Function DomainName( const AHost : String) : String
679: Function DownloadFile(SourceFile, DestFile: string): Boolean; //fast!
680: Function DownloadFileOpen(SourceFile, DestFile: string): Boolean; //open process
681: Function DosPathToUnixPath( const Path : string) : string
682: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
683: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
684: Function DoubleToBcd( const AValue : Double) : TBcd;
685: Function DoubleToHex( const D : Double) : string
686: Function DoUpdates : Boolean
687: Function Dragging: Boolean;
688: Function DrawCaption( pl : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
689: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL
690: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
691: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL
692: {Works like InputQuery but displays 2 edits. If PasswordChar <> #0, second edit's PasswordChar is set}
693: Function DualInputQuery(const ACapt,Prpt1,Prpt2:str;var AVall,AVal2:string;PasswordChar:Char=#0):Bool;
694: Function DupeString( const AText : string; ACount : Integer) : string
695: Function Edit : Boolean
696: Function EditCaption : Boolean
697: Function EditText : Boolean

```

```

698: Function EditFolderList( Folders : TStrings ) : Boolean
699: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext ) : Boolean
700: Function Elapsed( const Update : Boolean ) : Cardinal
701: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string ) : Boolean
702: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string ) : Boolean
703: Function EncodeDate( Year, Month, Day : Word ) : TDateTime
704: function EncodeDate(Year, Month, Day: Word): TDateTime;
705: Function EncodeDateDay( const AYear, ADayOfYear : Word ) : TDateTime
706: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word ) : TDateTime
707: Function EncodeDateTime( const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond: Word ): TDateTime
708: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word ) : TDateTime
709: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word ) : TDateTime
710: Function EncodeString( s : string ) : string
711: Function DecodeString( s : string ) : string
712: Function EncodeTime( Hour, Min, Sec, MSec : Word ) : TDateTime
713: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
714: Function EndIP : String
715: Function EndOfDay( const AYear, AMonth, ADay : Word ) : TDateTime;
716: Function EndOfDay1( const AYear, ADayOfYear : Word ) : TDateTime;
717: Function EndOfMonth( const AYear, AMonth : Word ) : TDateTime
718: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word ) : TDateTime
719: Function EndOfAYear( const AYear : Word ) : TDateTime
720: Function EndOfTheDay( const AValue : TDateTime ) : TDateTime
721: Function EndOfTheMonth( const AValue : TDateTime ) : TDateTime
722: Function EndOfTheWeek( const AValue : TDateTime ) : TDateTime
723: Function EndOfTheYear( const AValue : TDateTime ) : TDateTime
724: Function EndPeriod( const Period : Cardinal ) : Boolean
725: Function EndsStr( const ASubText, AText : string ) : Boolean
726: Function EndsText( const ASubText, AText : string ) : Boolean
727: Function EnsureMsgIDBrackets( const AMsgID : String ) : String
728: Function EnsureRange( const AValue, AMin, AMax : Integer ) : Integer;
729: Function EnsureRange1( const AValue, AMin, AMax : Int64 ) : Int64;
730: Function EnsureRange2( const AValue, AMin, AMax : Double ) : Double;
731: Function EOF: boolean
732: Function EOLn: boolean
733: Function EqualRect( const R1, R2 : TRect ) : Boolean
734: function EqualRect(const R1, R2: TRect): Boolean
735: Function Equals( Strings : TWideStrings ) : Boolean
736: function Equals(Strings: TStrings): Boolean;
737: Function EqualState( oState : ThRegularExpressionState ) : boolean
738: Function ErrOutput: Text
739: function ExceptionParam: String;
740: function ExceptionPos: Cardinal;
741: function ExceptionProc: Cardinal;
742: function ExceptionToString(er: TIFEException; Param: String): String;
743: function ExceptionType: TIFEException;
744: Function ExcludeTrailingBackslash( S : string ) : string
745: function ExcludeTrailingBackslash(const S: string): string
746: Function ExcludeTrailingPathDelimiter( const APath : string ) : string
747: Function ExcludeTrailingPathDelimiter( S : string ) : string
748: function ExcludeTrailingPathDelimiter(const S: string): string
749: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings) : boolean;
750: Function ExecProc : Integer
751: Function ExecSQL : Integer
752: Function ExecSQL( ExecDirect : Boolean ) : Integer
753: Function Execute : _Recordset;
754: Function Execute : Boolean
755: Function Execute : Boolean;
756: Function Execute( const SQL : string; Params : TParams; Cache : Boolean; cursor : phDBICur ) : Integer
757: Function Execute( const SQL : WideString; Params : TParams; ResultSet : TPSResult ) : Integer
758: Function Execute( ParentWnd : HWND ) : Boolean
759: Function Execute(constCommText:WideString;const CType:TCommType;const ExecuteOpts:TExecuteOpts):_Recordset;
760: Function Execute1( const Parameters : OleVariant ) : _Recordset;
761: Function Execute1( ParentWnd : HWND ) : Boolean;
762: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant ) : _Recordset;
763: Function ExecuteAction( Action : TBasicAction ) : Boolean
764: Function ExecuteDirect( const SQL : WideString ) : Integer
765: Function ExecuteFile( const FileName:string;const Params:string;const DefDir:string;ShowCmd:Int ):THandle
766: Procedure ExecuteThread2(func:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure'
767: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle
768: function ExeFileIsRunning(ExeFile: string): boolean;
769: function ExePath: string;
770: function ExePathName: string;
771: Function Exists( AItem : Pointer ) : Boolean
772: Function ExitWindows( ExitCode : Cardinal ) : Boolean
773: function Exp(x: Extended) : Extended;
774: Function ExpandEnvironmentVar( var Value : string ) : Boolean
775: Function ExpandFileName( FileName : string ) : string
776: function ExpandFileName(const FileName: string): string
777: Function ExpandUNCFileName( FileName : string ) : string
778: function ExpandUNCFileName(const FileName: string): string
779: Function ExpJ( const X : Float ) : Float;
780: Function Exsecans( X : Float ) : Float
781: Function Extract( const AByteCount : Integer ) : string
782: Function Extract( Item : TClass ) : TClass
783: Function Extract( Item : TComponent ) : TComponent
784: Function Extract( Item : TObject ) : TObject
785: Function ExtractFileDir( FileName : string ) : string

```

```

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

```

```

875: Function FindCdLineSwitch( Switch : string; IgnoreCase : Boolean) : Boolean;
876: Function FindCloseW(FindFile: Integer): LongBool; stdcall;
877: Function FindCmdLineSwitch( Switch : string; Chars : TSysCharSet; IgnoreCase : Boolean) : Boolean;
878: Function FindCmdLineSwitch( Switch : string) : Boolean;
879: function FindComponent(AName: String): TComponent;
880: function FindComponent(vlabel: string): TComponent;
881: function FindComponent2(vlabel: string): TComponent;
882: function FindControl(Handle: HWnd): TWinControl;
883: Function FindData( StartIndex : Integer; Value : Pointer; Inclusive, Wrap : Boolean) : TListItem;
884: Function FindDatabase( const DatabaseName : string) : TDatabase;
885: function FindDragTarget(const Pos: TPoint; AllowDisabled: Boolean): TControl;
886: Function FINDFIELD( const FIELDNAME : STRING) : TFIELD;
887: Function FINDFIELD( const FIELDNAME : WideString) : TFIELD;
888: Function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer;
889: Function FindNext2(var F: TSearchRec): Integer;
890: procedure FindClose2(var F: TSearchRec);
891: Function FINDFIRST : BOOLEAN;
892: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
893:   sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms, sfRecent, sfSendTo,
894:   sfStartMenu, stStartUp, sfTemplates);
895: FFolder: array [TJvSpecialFolder] of Integer =
896:   (CSIDL_BITBUCKET, CSDL_CONTROLS, CSDL_DESKTOP, CSDL_DESKTOPDIRECTORY,
897:    CSDL_DRIVES, CSDL_FONTS, CSDL_NETHOOD, CSDL_NETWORK, CSDL_PERSONAL,
898:    CSDL_PRINTERS, CSDL_PROGRAMS, CSDL_RECENT, CSDL_SENDTO, CSDL_STARTMENU,
899:    CSDL_STARTUP, CSDL_TEMPLATES);
900: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean;
901: function FindFirst(const filepath: string; attr: integer): integer;
902: function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer;
903: Function FindFirstOf( AFind, AText : String) : Integer;
904: Function FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState;
905: Function FINDINDEXFORFIELDS( const FIELDS : String) : TINDEXDEF;
906: Function FindInstanceOf( AClass : TClass; AExact : Boolean; AStartAt : Integer) : Integer;
907: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND) : TMENUITEM;
908: function FindItemId( Id : Integer) : TCollectionItem;
909: Function FindKey( const KeyValues : array of const) : Boolean;
910: Function FINDLAST : BOOLEAN;
911: Function FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl;
912: Function FindModuleClass( AClass : TComponentClass) : TComponent;
913: Function FindModuleName( const AClass : string) : TComponent;
914: Function FINDNEXT : BOOLEAN;
915: function FindNext: integer;
916: function FindNext2(var F: TSearchRec): Integer;
917: Function FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible : Boolean) : TTabSheet;
918: Function FindNextToSelect : TTreeNode;
919: Function FINDPARAM( const VALUE : String) : TPARAM;
920: Function FindParam( const Value : WideString) : TParameter;
921: Function FINDPRIOR : BOOLEAN;
922: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar) : TResourceHandle;
923: Function FindSession( const SessionName : string) : TSession;
924: function FindStringResource(Ident: Integer): string;
925: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer;
926: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString) : AnsiString;
927: function FindVCLWindow(const Pos: TPoint): TWinControl;
928: function FindWindow(C1, C2: PChar): Longint;
929: Function FindInPaths(const fileName,paths: String): String;
930: Function Finger : String;
931: Function First : TClass;
932: Function First : TComponent;
933: Function First : TObject;
934: Function FirstDelimiter( const delimiters : string; const Str : String) : integer;
935: Function FirstDelimiterI( const delimiters : WideString; const Str : WideString) : integer;
936: Function FirstInstance( const ATitle : string) : Boolean;
937: Function FloatPoint( const X, Y : Float) : TFloatPoint;
938: Function FloatPointI( const P : TPoint) : TFloatPoint;
939: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) : Boolean;
940: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
941: Function FloatRectI( const Rect : TRect) : TFloatRect;
942: Function FloatsEqual( const X, Y : Float) : Boolean;
943: Function FloatToBin(const D: Double): string; //doubletohex -> hextobin! in buffer;
944: Function FloatToCurr( Value : Extended) : Currency;
945: Function FloatToDateTime( Value : Extended) : TDateTime;
946: Function FloatToStr( Value : Extended) : string;
947: Function FloatToStr(e : Extended) : String;
948: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer) : string;
949: function FloatToStrF(Value: Extended; Format: TFloatFormat;Precision:Integer; Digits: Integer): string;
950: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings : TFormatSettings) : string;
951: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings : TFormatSettings) : string;
952: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue;Format: TFloatFormat; Precision,Digits: Integer): Integer;
953: Function Floor( const X : Extended) : Integer;
954: Function FloorInt( Value : Integer; StepSize : Integer) : Integer;
955: Function FloorJ( const X : Extended) : Integer;
956: Function Flush( const Count : Cardinal) : Boolean;
957: Function Flush(var t: Text): Integer;
958: function FmtLoadStr(Ident: Integer; const Args: array of const): string;
959: function FOCUSED:BOOLEAN

```

```

960: Function ForceBackslash( const PathName : string ) : string
961: Function ForceDirectories( const Dir : string ) : Boolean
962: Function ForceDirectories( Dir : string ) : Boolean
963: Function ForceDirectories( Name : string ) : Boolean
964: Function ForceInBox( const Point : TPoint; const Box : TRect ) : TPoint
965: Function ForceInRange( A, Min, Max : Integer ) : Integer
966: Function ForceInRangeR( const A, Min, Max : Double ) : Double
967: Function ForEach( AProc : TBucketProc; AInfo : Pointer ) : Boolean;
968: Function ForEach1( AEvent : TBucketEvent ) : Boolean;
969: Function ForegroundTask: Boolean
970: function Format( const Format: string; const Args: array of const): string;
971: Function FormatBcd( const Format : string; Bcd : TBcd ) : string
972: FUNCTION FormatBigInt(s: string): STRING;
973: function FormatBytesize(const bytes: int64): string;
974: function FormatBuf(var Buffer:PChar;BufLen:Card;const Format:string;FmtLen:Card;const Args:array of const):Cardinal
975: Function FormatCount( iCount : integer; const sSingular : string; const sPlural : string ) : string
976: Function FormatCurr( Format : string; Value : Currency) : string;
977: function FormatCurr(const Format: string; Value: Currency): string
978: Function FormatDateTime( Format : string; DateTime : TDateTime ) : string;
979: function FormatDateTime(const fmt: string; D: TDateTime): string;
980: Function FormatFloat( Format : string; Value : Extended) : string;
981: function FormatFloat(const Format: string; Value: Extended): string
982: Function FormatFloat( Format : string; Value : Extended) : string;
983: Function FormatFloat2( Format : string; Value : Extended; FormatSettings : TFormatSettings ) : string;
984: Function FormatCurr( Format : string; Value : Currency) : string;
985: Function FormatCurr2(Format: string; Value : Currency; FormatSettings : TFormatSettings) : string;
986: Function Format2(const Format:string;const Args:array of const;const FSettings:TFormatSettings): string
987: FUNCTION FormatInt(i: integer): STRING;
988: FUNCTION FormatInt64(i: int64): STRING;
989: Function FormatMaskText( const EditMask : string; const Value : string ) : string
990: Function FormatValue( AValue : Cardinal ) : string
991: Function FormatVersionString( const HiV, LoV : Word ) : string;
992: Function FormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
993: function Frac(X: Extended): Extended;
994: Function FreeResource( ResData : HGLOBAL ) : LongBool
995: Function FromCommon( const AValue : Double) : Double
996: function FromCommon(const AValue: Double): Double;
997: Function FTPGMT.ToDateTimeToMLS( const ATimestamp : TDateTime; const AIncludeMSecs : Boolean) : String
998: Function FTPLocalDateTimeToMLS( const ATimestamp : TDateTime; const AIncludeMSecs : Boolean) : String
999: Function FTPMLSTOGMTDateTime( const ATimestamp : String) : TDateTime
1000: Function FTPMLSTOLocalDateTime( const ATimestamp : String) : TDateTime
1001: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
1002: //Function Funclist Size is: 6444 of mx3.9.8.9
1003: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:TPaymentTime):Extended
1004: Function FulltimeToStr(SUMTime: TDateTime): string;';
1005: Function Gauss( const x, Spread : Double ) : Double
1006: function Gauss(const x,Spread: Double): Double;
1007: Function GCD(x, y : LongInt) : LongInt;
1008: Function GCDJ( X, Y : Cardinal ) : Cardinal
1009: Function GDAL: LongWord
1010: Function GdiFlush : BOOL
1011: Function GdiSetBatchLimit( Limit : DWORD ) : DWORD
1012: Function GdiGetBatchLimit : DWORD
1013: Function GenerateHeader : TIHeaderList
1014: Function GeometricMean( const X : TDynFloatArray) : Float
1015: Function Get( AURL : string) : string;
1016: Function Get2( AURL : string) : string;
1017: Function Get8087CW : Word
1018: function GetActiveOleObject(const ClassName: String): IDispatch;
1019: Function GetAliasDriverName( const AliasName : string ) : string
1020: Function GetAPMBatteryFlag : TAPMBatteryFlag
1021: Function GetAPMBatteryFullLifeTime : DWORD
1022: Function GetAPMBatteryLifePercent : Integer
1023: Function GetAPMBatteryLifeTime : DWORD
1024: Function GetAPMLineStatus : TAPMLineStatus
1025: Function GetAppdataFolder : string
1026: Function GetAppDispatcher : TComponent
1027: function GetArrayLength: integer;
1028: Function GetASCII: string;
1029: Function GetASCIILine: string;
1030: Function GetAsHandle( Format : Word) : THandle
1031: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1032: Function GetBackupFileName( const FileName : string ) : string
1033: function GetBaseAddress(PID:DWORD):DWORD; //Process API
1034: Function GetBBitmap( Value : TBitmap ) : TBitmap
1035: Function GetBIOSCopyright : string
1036: Function GetBIOSDate : TDateTime
1037: Function GetBIOSExtendedInfo : string
1038: Function GetBIOSName : string
1039: Function getBitmap(apath: string): TBitmap;
1040: Function GetBitmap( Index : Integer; Image : TBitmap) : Boolean //object
1041: Function getBitMapObject(const bitmappath: string): TBitmap;
1042: Function GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1043: Function GetCapsLockKeyState : Boolean
1044: function GetCaptureControl: TControl;
1045: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char) : TJclCdTrackInfo;
1046: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char) : string;

```

```

1047: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) : string
1048: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char) : string
1049: Function GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1050: Function GetClockValue : Int64
1051: function getCmdLine: PChar;
1052: function getCmdShow: Integer;
1053: function GetCPUSpeed: Double;
1054: Function GetColField( DataCol : Integer) : TField
1055: Function GetColorBlue( const Color : TColor) : Byte
1056: Function GetColorFlag( const Color : TColor) : Byte
1057: Function GetColorGreen( const Color : TColor) : Byte
1058: Function GetColorRed( const Color : TColor) : Byte
1059: Function GetComCtlVersion : Integer
1060: Function GetComPorts: TStringlist;
1061: Function GetCommonAppdataFolder : string
1062: Function GetCommonDesktopdirectoryFolder : string
1063: Function GetCommonFavoritesFolder : string
1064: Function GetCommonFilesFolder : string
1065: Function GetCommonProgramsFolder : string
1066: Function GetCommonStartmenuFolder : string
1067: Function GetCommonStartupFolder : string
1068: Function GetComponent( Owner, Parent : TComponent) : TComponent
1069: Function GetConnectionRegistryFile( DesignMode : Boolean) : string
1070: Function GetCookiesFolder : string
1071: Function GetCPUSpeed( var CpuSpeed : TFreqInfo) : Boolean
1072: Function GetCurrent : TFavoriteLinkItem
1073: Function GetCurrent : TListItem
1074: Function GetCurrent : TTaskDialogBaseButtonItem
1075: Function GetCurrent : TToolButton
1076: Function GetCurrent : TTreeNode
1077: Function GetCurrent : WideString
1078: Function GetCurrentDir : string
1079: function GetCurrentDir: string)
1080: Function GetCurrentFolder : string
1081: Function GETCURRENTRECORD( BUFFER : PCHAR) : BOOLEAN
1082: Function GetCurrentProcessId : TIdPID
1083: Function GetCurrentThreadHandle : THandle
1084: Function GetCurrentThreadId: LongWord; stdcall;
1085: Function GetCustomHeader( const Name : string) : String
1086: Function GetDataItem( Value : Pointer) : Longint
1087: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string) : Integer;
1088: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string) : Integer;
1089: Function GETDATASIZE : INTEGER
1090: Function GetDC(hwnd: HWND) : HDC;
1091: Function GetDefaultFileExt( const MIMEType : string) : string
1092: Function GetDefaults : Boolean
1093: Function GetDefaultSchemaName : WideString
1094: Function GetDefaultStreamLoader : IStreamLoader
1095: Function GetDesktopDirectoryFolder : string
1096: Function GetDesktopFolder : string
1097: Function GetDFAState( oStates : Tlist) : TniRegularExpressionState
1098: Function GetDirectorySize( const Path : string) : Int64
1099: Function GetDisplayWidth : Integer
1100: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer) : Boolean
1101: Function GetDomainName : string
1102: Function GetDriverRegistryFile( DesignMode : Boolean) : string
1103: function GetDriveType(rootpath: pchar): cardinal;
1104: Function GetDriveTypeStr( const Drive : Char) : string
1105: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1106: Function GetEnumerator : TListItemsEnumerator
1107: Function GetEnumerator : TTaskDialogButtonsEnumerator
1108: Function GetEnumerator : TToolBarEnumerator
1109: Function GetEnumerator : TTreeNodesEnumerator
1110: Function GetEnumerator : TWideStringsEnumerator
1111: Function GetEnvVar( const VarName : string) : string
1112: Function GetEnvironmentVar( const AVariableName : string) : string
1113: Function GetEnvironmentVariable( const VarName : string) : string
1114: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean) : Boolean
1115: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean) : Boolean
1116: Function getEnvironmentString: string;
1117: Function GetExceptionHandler : TObject
1118: Function GetFavoritesFolder : string
1119: Function GetFieldByName( const Name : string) : string
1120: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo) : Boolean
1121: Function GetFieldValue( ACol : Integer) : string
1122: Function GetFileAgeCoherence( const FileName : string) : Boolean
1123: Function GetFileCreation( const FileName : string) : TFileTime
1124: Function GetFileCreationTime( const Filename : string) : TDateTime
1125: Function GetFileInfo( const FileName : string) : TSearchRec
1126: Function GetFileLastAccess( const FileName : string) : TFileTime
1127: Function GetFileLastWrite( const FileName : string) : TFileTime
1128: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1129: Function GetFileList1(apath: string): TStringlist;
1130: Function GetFileMIMETYPE( const AFileName : string) : string
1131: Function GetFileSize( const FileName : string) : Int64
1132: Function GetFileVersion( AFileName : string) : Cardinal
1133: Function GetFileVersion( const Afilename : string) : Cardinal
1134: Function GetFileSize2(Handle: Integer; x: Integer): Integer; stdcall;
1135: Function GetFileDate(aFile:string; aWithTime:Boolean):string;

```

```

1136: Function GetFileCount(adirmask: string): integer; //files count in directory!
1137: Function GetFilterData( Root : PExprNode ) : TExprData
1138: Function getFirstChild : LongInt
1139: Function getChild : TTreeNode
1140: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string ) : string
1141: Function GetFirstNode : TTreeNode
1142: Function GetFontsFolder : string
1143: Function GetFormulaValue( const Formula : string ) : Extended
1144: Function GetFreePageFileMemory : Integer
1145: Function GetFreePhysicalMemory : Integer
1146: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind ) : Integer;
1147: Function GetFreeSystemResources1 : TFreeSystemResources;
1148: Function GetFreeVirtualMemory : Integer
1149: Function GetFromClipboard : Boolean
1150: Function GetFullURI( const AOptionalFields : TIdURIOptionalFieldsSet ) : String
1151: Function GetGBTmap( Value : TBitmap ) : TBitmap
1152: Function GetGMTDateByName( const AFileName : TIdFileName ) : TDateTime
1153: Function GetGroupState( Level : Integer ) : TGroupPosInds
1154: Function GetHandle : HWND
1155: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN ) : THELPCONTEXT
1156: function GetHexArray(ahexdig: THexArray): THexArray;
1157: Function GetHighlightColor( const Color : TColor; Luminance : Integer ) : TColor
1158: function GetHINSTANCE: longword;
1159: Function GetHistoryFolder : string
1160: Function GetHitTestInfoAt( X, Y : Integer ) : THitTests
1161: function getHMODULE: longword;
1162: Function GetHostName( const AComputerName: String): String;
1163: Function GetHostName : string
1164: Function getHOSTIP: string;
1165: Function GetHotSpot : TPoint
1166: Function GetHueBitmap( Value : TBitmap ) : TBitmap
1167: Function GetImageBitmap : HBITMAP
1168: Function GETIMAGELIST : TCUSTOMIMAGELIST
1169: Function GetIncome( const aNetto : Currency ) : Currency
1170: Function GetIncome( const aNetto : Extended ) : Extended
1171: Function GetIncome( const aNetto : Extended ) : Extended
1172: Function GetIncome( const aNetto : Extended ) : Extended
1173: function GetIncome( const aNetto : Currency ) : Currency
1174: Function GetIncome2( const aNetto : Currency ) : Currency
1175: Function GetIncome2( const aNetto : Currency ) : Currency
1176: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string ) : string
1177: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN ) : TINDEXDEF
1178: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet ) : TIndexDef
1179: Function GetInstRes(Instance:THandle;ResType:TResType;const
  Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1180: Function
  GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1181: Function GetIntelCacheDescription( const D : Byte ) : string
1182: Function GetInteractiveUserName : string
1183: Function GetInternetCacheFolder : string
1184: Function GetInternetFormattedFileTimeStamp( const AFilename : String ) : String
1185: Function GetIPAddress( const HostName : string ) : string
1186: Function GetIP( const HostName : string ) : string
1187: Function GetIPHostByName(const AComputerName: String): String;
1188: Function GetIsAdmin: Boolean;
1189: Function GetItem( X, Y : Integer ) : LongInt
1190: Function GetItemAt( X, Y : Integer ) : TListItem
1191: Function GetItemHeight(Font: TFont): Integer;
1192: Function GetItemPath( Index : Integer ) : string
1193: Function GetKeyFieldNames( List : TStrings ) : Integer;
1194: Function GetKeyFieldNames1( List : TWideStrings ) : Integer;
1195: Function GetKeyState( const VirtualKey : Cardinal ) : Boolean
1196: Function GetLastChild : LongInt
1197: Function GetLastChild : TTreeNode
1198: Function GetLastDelimitedToken( const cDelim : char; const cStr : string ) : string
1199: function GetLastError: Integer
1200: Function GetLAT_CONV_FACTOR: double; //for WGS84 power(1 - 1 / 298.257223563, 2);
1201: Function GetLinesCount(sfileName : String): Integer;
1202: Function GetLoader( Ext : string ) : TBitmapLoader
1203: Function GetLoadFilter : string
1204: Function GetLocalComputerName : string
1205: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char ) : Char
1206: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string ) : string
1207: Function GetLocalUserName : string
1208: Function GetLoginUsername : WideString
1209: function getLongDayNames: string)
1210: Function GetLongHint( const hint: string): string
1211: function getLongMonthNames: string)
1212: Function GetMacAddresses( const Machine : string; const Addresses : TStrings ) : Integer
1213: Function GetMainAppWndFromPid( PID : DWORD ) : HWND
1214: Function GetMapX(C_form,apath: string; const Data: string): boolean; //c_form: [html/json/xml]
1215: //if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
1216: Procedure GetGRMap(C_form,apath: string; const Data: string);
1217: Function GetMapXGeoReverse(C_form: string; const lat,long: string): string;
1218: //if GetMapXGeoReverse('XML',topPath,'47.0397826','7.62914761277888') then
1219: Function GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
1220: Function GetMaskBitmap : HBITMAP
1221: Function GetMaxAppAddress : Integer
1222: Function GetMciErrorMessage( const MciErrNo : MCIERROR ) : string

```

```

1223: Function GetMemoryLoad : Byte
1224: Function GetMIMEDefaultFileExt( const MIMEType : string ) : TIdFileName
1225: Function GetMIMETypeFromFile( const AFile : string ) : string
1226: Function GetMIMETypeFromFile( const AFile : TIdFileName ) : string
1227: Function GetMinAppAddress : Integer
1228: Function GetModule : TComponent
1229: Function GetModuleHandle( ModuleName : PChar ) : HMODULE
1230: Function GetModuleName( Module : HMODULE ) : string
1231: Function GetModulePath( const Module : HMODULE ) : string
1232: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1233: Function GetMorseID(InChar : Char): Word;')
1234: Function GetMorseString2(InChar : Char): string;');
1235: Function GetMorseLine(dots: boolean): string;') //whole table! {1 or dots}
1236: Function GetMorseTable(dots: boolean): string;') //whole table!
1237: Function GetMorseSign(InChar : Char): string;');
1238: Function GetCommandLine: PChar; stdcall;
1239: Function GetMonochromeBitmap( Value : TBitmap ) : TBitmap
1240: Function GetMultiN(aval: integer): string;
1241: Function GetName : String
1242: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection ) : TListItem
1243: Function GetNethoodFolder : string
1244: Function GetNext : TTreeNode
1245: Function GetNextChild( Value : LongInt ) : LongInt
1246: Function GetNextChild( Value : TTreeNode ) : TTreeNode
1247: Function GetNextDelimitedToken( const cDelim : char; var cStr : String ) : String
1248: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates ) : TListItem
1249: Function GetNextPacket : Integer
1250: Function getNextSibling : TTreeNode
1251: Function GetNextVisible : TTreeNode
1252: Function GetNode( ItemId : HTreeItem ) : TTreeNode
1253: Function GetNodeAt( X, Y : Integer ) : TTreeNode
1254: Function GetNodeDisplayWidth( Node : TOutlineNode ) : Integer
1255: function GetNumberOfProcessors: longint;
1256: Function GetNumLockKeyState : Boolean
1257: Function GetObjectProperty( Instance : TPersistent; const PropName : string ) : TObject
1258: Function GetOnlyTransitionOn( cChar : char ) : TniRegularExpressionState
1259: Function GetOptionalParam( const ParamName : string ) : OleVariant
1260: Function GetOSName: string;
1261: Function GetOSVersion: string;
1262: Function GetOsNumber: string;
1263: Function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
1264: Function GetPackageModuleHandle( PackageName : PChar ) : HMODULE
1265: function GetPageSize: Cardinal;
1266: Function GetParameterFileName : string
1267: Function GetParams( var OwnerData : OleVariant ) : OleVariant
1268: Function GETPARENTCOMPONENT : TCOMPONENT
1269: Function GetParentForm(control: TControl): TForm
1270: Function GETPARENTMENU : TMENU
1271: Function GetPassword : Boolean
1272: Function GetPassword : string
1273: Function GetPersonalFolder : string
1274: Function GetPidFromProcessName( const ProcessName : string ) : DWORD
1275: function getPI: extended; //of const PI math
1276: Function GetPosition : TPoint
1277: Function GetPrev : TTreeNode
1278: Function GetPrevChild( Value : LongInt ) : LongInt
1279: Function GetPrevChild( Value : TTreeNode ) : TTreeNode
1280: Function getPrevSibling : TTreeNode
1281: Function GetPrevVisible : TTreeNode
1282: Function GetPrinthoodFolder : string
1283: Function GetPrivilegeDisplayName( const PrivilegeName : string ) : string
1284: Function getProcessList: TString;
1285: Function GetProcessId : TIdPID
1286: Function GetProcessNameFromPid( PID : DWORD ) : string
1287: Function GetProcessNameFromWnd( Wnd : HWND ) : string
1288: Function GetProcessMemoryInfo(Process: THandle;ppsmemCounters: TProcessMemoryCounters;cb: DWORD):BOOL
1289: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1290: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1291: Function GetProgramFilesFolder : string
1292: Function GetProgramsFolder : string
1293: Function GetProxy : string
1294: Function GetQuoteChar : WideString
1295: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const Data:string;aformat:string):TLinearBitmap;
1296: Function GetQrCodetoFile(Width,Height:Word;Correct_Level:string;const Data:string;aformat:string):TLinearBitmap;
1297: Function GetRate : Double
1298: Function getPerfTime: string;
1299: Function getRuntime: string;
1300: Function GetRBitmap( Value : TBitmap ) : TBitmap
1301: Function GetReadableName( const AName : string ) : string
1302: Function GetRecentDocs : TStringList
1303: Function GetRecentFolder : string
1304: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer ) : OleVariant;
1305: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var Params, OwnerData : OleVariant);
1306: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString ) : _Recordset
1307: Function GetRegisteredCompany : string
1308: Function GetRegisteredOwner : string

```

```

1309: Function GetResource(ResType:TResType;const
  Name:string;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor:Bool
1310: Function GetResourceName( ObjStream : TStream; var AName : string ) : Boolean
1311: Function GetResponse( const AAllowedResponses : array of SmallInt ) : SmallInt;
1312: Function GetResponse( const AAllowedResponse : SmallInt ) : SmallInt;
1313: Function GetRValue( rgb : DWORD) : Byte
1314: Function GetGValue( rgb : DWORD) : Byte
1315: Function GetBValue( rgb : DWORD) : Byte
1316: Function GetCValue( cmyk : COLORREF) : Byte
1317: Function GetMValue( cmyk : COLORREF) : Byte
1318: Function GetYValue( cmyk : COLORREF) : Byte
1319: Function GetKValue( cmyk : COLORREF) : Byte
1320: Function CMYK( c, m, y, k : Byte) : COLORREF
1321: Procedure GetScreenshot(var ABitmap : TBitmap);
1322: Function GetOSName: string;
1323: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR ) : FARPROC
1324: Function GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1325: Function GetSafeCallExceptionMsg : String
1326: Function GetSaturationBitmap( Value : TBitmap ) : TBitmap
1327: Function GetSaveFilter : string
1328: Function GetSaver( Ext : string ) : TBitmapLoader
1329: Function GetScrollLockKeyState : Boolean
1330: Function GetSearchString : string
1331: Function GetSelections( Alist : TList ) : TTreeNode
1332: function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1333: Function GetSendToFolder : string
1334: Function GetServer : IAppServer
1335: Function GetServerList : OleVariant
1336: Function GetShadowColor( const Color : TColor; Luminance : Integer ) : TColor
1337: Function GetShellProcessHandle : THandle
1338: Function GetShellProcessName : string
1339: Function GetShellVersion : Cardinal
1340: function getShortDayNames: string)
1341: Function GetShortHint( const hint: string): string
1342: function getShortMonthNames: string)
1343: Function GetSizeOfFile( const FileName : string ) : Int64;
1344: Function GetSizeOfFile1( Handle : THandle ) : Int64;
1345: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1346: Function GetStartmenuFolder : string
1347: Function GetStartupFolder : string
1348: Function GetStringProperty( Instance : TPersistent; const PropName : string ) : WideString
1349: Function GetSuccessor( cChar : char ) : TniRegularExpressionState
1350: Function GetSwapFileSize : Integer
1351: Function GetSwapFileUsage : Integer
1352: Function GetSystemLocale : TIdCharSet
1353: Function GetSystemMetrics( nIndex : Integer ) : Integer
1354: Function GetSystemPathSH(Folder: Integer): TFilename ;
1355: Function GetTableNameFromQuery( const SQL : Widestring ) : Widestring
1356: Function GetTableNameFromSQL( const SQL : WideString ) : WideString
1357: Function GetTableNameFromSQLEX( const SQL : WideString; IdOption : IDENTIFEROPTION ) : WideString
1358: Function GetTasksList( const List : TStrings ) : Boolean
1359: Function getTeamViewerID: string;
1360: Function GetTemplatesFolder : string
1361: Function GetText : PwideChar
1362: function GetText:PChar
1363: Function GetTextBuf( Buffer : PChar; BufSize : Integer ) : Integer
1364: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1365: Function GetTextItem( const Value : string ) : Longint
1366: function GETTEXTLEN:INTEGER
1367: Function GetThreadLocale: Longint; stdcall
1368: Function GetCurrentThreadID: LongWord; stdcall;
1369: Function GetTickCount : Cardinal
1370: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal ) : Cardinal
1371: Function GetTicketNr : longint
1372: Function GetTime : Cardinal
1373: Function GetTime : TDateTime
1374: Function GetTimeout : Integer
1375: Function GetTimeStr: String
1376: Function GetTimeString: String
1377: Function GetTodayFiles(startdir, amask: string): TStringlist;
1378: Function getTokenCounts : integer
1379: Function GetTotalPageFileMemory : Integer
1380: Function GetTotalPhysicalMemory : Integer
1381: Function GetTotalVirtualMemory : Integer
1382: Function GetUniqueFileName( const APath, APrefix, AExt : String ) : String
1383: Function GetUseNowForDate : Boolean
1384: Function GetUserDomainName( const CurUser : string ) : string
1385: Function GetUserName: string;
1386: Function GetUserName: string;
1387: Function GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD) : BOOL';
1388: Function GetUserObjectName( hUserObject : THandle ) : string
1389: Function GetValueBitmap( Value : TBitmap ) : TBitmap
1390: Function GetValueMSec : Cardinal
1391: Function GetValueStr : String
1392: Function GetVersion: int;
1393: Function GetVersionString(FileName: string): string;
1394: Function getVideoDrivers: string;
1395: Function GetVisibleNode( Index : LongInt ) : TOutlineNode
1396: Function GetVolumeFileSystem( const Drive : string ) : string

```

```

1397: Function GetVolumeName( const Drive : string) : string
1398: Function GetVolumeSerialNumber( const Drive : string) : string
1399: Function GetWebAppServices : IWebAppServices
1400: Function GetWebRequestHandler : IWebRequestHandler
1401: Function GetWindowCaption( Wnd : HWND) : string
1402: Function GetWindowDC(hdwnd: HWND) : HDC;
1403: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1404: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1405: Function GetWindowsComputerID : string
1406: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1407: Function GetWindowsFolder : string
1408: Function GetWindowsServicePackVersion : Integer
1409: Function GetWindowsServicePackVersionString : string
1410: Function GetWindowsSystemFolder : string
1411: Function GetWindowsTempFolder : string
1412: Function GetWindowsUserID : string
1413: Function GetWindowsVersion : TWindowsVersion
1414: Function GetWindowsVersionString : string
1415: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1416: Function GMTToLocalDateTime( S : string) : TDateTime
1417: Function GotoKey : Boolean
1418: Function GradToCycle( const Grads : Extended) : Extended
1419: Function GradToDeg( const Grads : Extended) : Extended
1420: Function GradToDeg( const Value : Extended) : Extended;
1421: Function GradToDeg1( const Value : Double) : Double;
1422: Function GradToDeg2( const Value : Single) : Single;
1423: Function GradToRad( const Grads : Extended) : Extended
1424: Function GradToRad( const Value : Extended) : Extended;
1425: Function GradToRad1( const Value : Double) : Double;
1426: Function GradToRad2( const Value : Single) : Single;
1427: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1428: Function GreenComponent( const Color32 : TColor32) : Integer
1429: function GUIDToString(const GUID: TGUID): string)
1430: Function HandleAllocated : Boolean
1431: function HandleAllocated: Boolean;
1432: Function HandleRequest : Boolean
1433: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1434: Function HarmonicMean( const X : TDynFloatArray) : Float
1435: Function HasAsParent( Value : TTreenode) : Boolean
1436: Function HASCHILDDEFS : BOOLEAN
1437: Function HasCurValues : Boolean
1438: Function HasExtendCharacter( const s : UTF8String) : Boolean
1439: Function HasFormat( Format : Word) : Boolean
1440: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1441: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1442: Function HashValue(AStream: TStream): LongWord
1443: Function HashValue(AStream: TStream): Word
1444: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1445: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1446: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1447: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1448: Function HashValue16( const ASrc : string) : Word;
1449: Function HashValue16Stream( AStream : TStream) : Word;
1450: Function HashValue32( const ASrc : string) : LongWord;
1451: Function HashValue32Stream( AStream : TStream) : LongWord;
1452: Function HasMergeConflicts : Boolean
1453: Function hasMoreTokens : boolean
1454: Function HASPARENT : BOOLEAN
1455: function HasParent: Boolean
1456: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1457: Function HasUTF8BOM( S : TStream) : boolean;
1458: Function HasUTF8BOM1( S : AnsiString) : boolean;
1459: Function Havervine( X : Float) : Float
1460: Function Head( s : string; const subs : string; var tail : string) : string
1461: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1462: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1463: function HELPJUMP(JUMPID:STRING):BOOLEAN
1464: Function HeronianMean( const a, b : Float) : Float
1465: function HexToStr(Value: string): string;
1466: function HexToBin(Text,Buffer:PChar; Bufsize:Integer):Integer;
1467: function HexToBin2(HexNum: string): string;
1468: Function Hex.ToDouble( const Hex : string) : Double
1469: function HexToInt(hexnum: string): LongInt;
1470: function HexToStr(Value: string): string;
1471: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1472: function Hi(vdat: word): byte;
1473: function HiByte(W: Word): Byte)
1474: function High: Int64;
1475: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1476: function HINSTANCE: longword;
1477: function HiWord(l: DWORD): Word
1478: function HMODULE: longword;
1479: Function HourOf( const AValue : TDateTime) : Word
1480: Function HourOfTheDay( const AValue : TDateTime) : Word
1481: Function HourOfTheMonth( const AValue : TDateTime) : Word
1482: Function HourOfTheWeek( const AValue : TDateTime) : Word
1483: Function HourOfTheYear( const AValue : TDateTime) : Word
1484: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1485: Function HourSpan( const ANow, AThen : TDateTime) : Double

```

```

1486: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1487: Function HTMLDecode( const AStr : String) : String
1488: Function HTMLEncode( const AStr : String) : String
1489: Function HTMLEscape( const Str : string) : string
1490: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1491: Function HTTPDecode( const AStr : String) : string
1492: Function HTTPEncode( const AStr : String) : string
1493: Function Hypot( const X, Y : Extended) : Extended
1494: Function IBMax( n1, n2 : Integer) : Integer
1495: Function IBMin( n1, n2 : Integer) : Integer
1496: Function IBRandomString( iLength : Integer) : String
1497: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1498: Function IBStripString( st : String; CharsToStrip : String) : String
1499: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1500: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1501: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1502: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1503: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1504: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1505: Function RandomString( iLength : Integer) : String';
1506: Function RandomInteger( iLow, iHigh : Integer) : Integer';
1507: Function StripString( st : String; CharsToStrip : String) : String';
1508: FUNCTION Strip(const SubString: String; MainString: String): String;
1509: function StripTags(const S: string): string; //<'> of HTML
1510: function SizeToString(size : Int64; const unitStr : String) : String;
1511: FUNCTION NumertoString(No: Word): String;
1512: Function FormatIdentifier( Dialect : Integer; Value : String) : String';
1513: Function FormatIdentifierValue( Dialect : Integer; Value : String) : String';
1514: Function ExtractIdentifier( Dialect : Integer; Value : String) : String';
1515: Function QuoteIdentifier( Dialect : Integer; Value : String) : String';
1516: Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
1517: Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String);
1518: function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLOToken): TSQLOToken;
1519: Function IconToBitmap( Ico : HICON) : TBitmap
1520: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1521: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1522: function IdentToCharset(const Ident : string; var Charset: Longint): Boolean
1523: function IdentToColor(const Ident: string; var Color: Longint): Boolean
1524: function IdentToCursor(const Ident : string; var cursor: Longint): Boolean;
1525: Function IdGetDefaultCharSet : TIDCharSet
1526: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1527: Function IdPorts2 : TStringList
1528: Function IdToMib( const Id : string) : string
1529: Function IdSHA1Hash(apath: string): string;
1530: Function IdHashSHA1(apath: string): string;
1531: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1532: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1533: Function IfThenInt( AValue : Boolean; const ATrue : integer; AFalse : integer): integer;';
1534: Function IfThenDouble( AValue : Boolean; const ATrue : double; AFalse : double): double;';
1535: Function IfThenBool( AValue : Boolean; const ATrue : boolean; AFalse : boolean): boolean;';
1536: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1537: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string): string;
1538: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean): Boolean;
1539: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1540: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1541: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1542: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1543: Function IncLimit1(var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1544: Function IncLimit2(var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1545: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1546: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1547: Function IncLimit5(var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1548: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1549: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1550: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1551: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1552: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1553: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1554: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1555: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1556: Function IncludeTrailingBackslash( S : string) : string
1557: function IncludeTrailingBackslash(const S: string): string
1558: Function IncludeTrailingPathDelimiter( const APPath : string) : string
1559: Function IncludeTrailingPathDelimiter( S : string) : string
1560: function IncludeTrailingPathDelimiter(const S: string): string
1561: Function IncludeTrailingSlash( const APPath : string) : string
1562: Function IncMillisecond( const AValue : TDateTime; const ANumberOfMilliseconds : Int64) : TDateTime
1563: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1564: Function IncMonth( const DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1565: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime
1566: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1567: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1568: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1569: Function IndexOf( AClass : TClass) : Integer
1570: Function IndexOf( AComponent : TComponent) : Integer
1571: Function IndexOf( AObject : TObject) : Integer
1572: Function INDEXOF( const ANAME : String) : INTEGER
1573: Function IndexOf( const DisplayName : string) : Integer
1574: Function IndexOf( const Item : TBookmarkStr) : Integer

```

```

1575: Function IndexOf( const S : WideString) : Integer
1576: Function IndexOf( const View : TJclFileMappingView) : Integer
1577: Function INDEXOF( FIELD : TFIELD) : INTEGER
1578: Function IndexOf( ID : LCID) : Integer
1579: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1580: Function IndexOf( Value : TListItem) : Integer
1581: Function IndexOf( Value : TTreeNode) : Integer
1582: function IndexOf(const S: string): Integer;
1583: Function IndexOfName( const Name : WideString) : Integer
1584: function IndexOfName(Name: string): Integer;
1585: Function IndexOfObject( AObject : TObject) : Integer
1586: function IndexOfObject(AObject:tObject):Integer
1587: Function IndexOfTabAt( X, Y : Integer) : Integer
1588: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1589: Function IndexText( const AText : string; const AValues : array of string) : Integer
1590: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer
1591: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer
1592: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer
1593: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1594: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1595: Function IndyGetHostName : string
1596: Function IndyInterlockedDecrement( var I : Integer) : Integer
1597: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1598: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1599: Function IndyInterlockedIncrement( var I : Integer) : Integer
1600: Function IndyLowerCase( const A1 : string) : string
1601: Function IndyStrToBool( const AString : String) : Boolean
1602: Function IndyUpperCase( const A1 : string) : string
1603: Function InitCommonControl( CC : Integer) : Boolean
1604: Function InitTempPath : string
1605: Function InMainThread : boolean
1606: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1607: Function Input: Text)
1608: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1609: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1610: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1611: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1612: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean
1613: Function InquireSignal( RtlSignalNum : Integer) : TSignalState
1614: Function InRanger( const A, Min, Max : Double) : Boolean
1615: function Insert( Index : Integer) : TCollectionItem
1616: Function Insert( Index : Integer) : TComboExItem
1617: Function Insert( Index : Integer) : THeaderSection
1618: Function Insert( Index : Integer) : TListItem
1619: Function Insert( Index : Integer) : TStatusPanel
1620: Function Insert( Index : Integer) : TWorkArea
1621: Function Insert( Index : Longint; const Text : string) : LongInt
1622: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1623: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1624: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1625: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1626: Function InsertObject( Index : Longint; const Text : string; const Data : Pointer) : LongInt
1627: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1628: Function Instance : Longint
1629: function InstanceSize : Longint
1630: Function Int(e : Extended) : Extended;
1631: function Int64ToStr(i: Int64): String;
1632: Function IntegerToBcd( const AValue : Integer) : TBcd
1633: Function Intensity( const Color32 : TColor32) : Integer;
1634: Function Intensity( const R, G, B : Single) : Single;
1635: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
FutureValue:Extended; PaymentTime : TPMT) : Extended
1636: Function InterestRate(NPeriods:Integer;const Payment,PresVal,
FutureVal:Extended;PaymentTime:TPMT):Extended
1637: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1638: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1639: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1640: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1641: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1642: Function IntMibToStr( const Value : string) : string
1643: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1644: Function IntToBin( Value : cardinal) : string
1645: Function IntToHex( Value : Integer; Digits : Integer) : string;
1646: function IntToHex(a: integer; b: integer): string;
1647: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1648: function IntToHex64(Value: Int64; Digits: Integer): string)
1649: Function IntTo3Str( Value : Longint; separator: string) : string
1650: Function inttobool( alnt : LongInt) : Boolean
1651: function IntToStr(i: Int64): String;
1652: Function IntToStr64(Value: Int64): string)
1653: function IOResult: Integer
1654: Function IPv6AddressToStr(const AValue: TIidIPv6Address) : string
1655: function IPAddrToHostName(const IP: string): string;
1656: Function IsAccel(VK: Word; const Str: string): Boolean
1657: Function IsAddressInNetwork( Address : String) : Boolean
1658: Function IsAdministrator : Boolean
1659: Function IsAlias( const Name : string) : Boolean
1660: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1661: Function IsASCII( const AByte : Byte) : Boolean;

```

```

1662: Function IsASCIIILDH( const AByte : Byte) : Boolean;
1663: Function IsAssembly(const FileName: string): Boolean;
1664: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1665: Function IsBinary(const AChar : Char) : Boolean
1666: function IsConsole: Boolean)
1667: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1668: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1669: Function IsDelphiDesignMode : boolean
1670: Function IsDelphiRunning : boolean
1671: Function IsDFAState : boolean
1672: Function IsDirectory( const FileName : string) : Boolean
1673: Function IsDomain( const S : String) : Boolean
1674: function IsDragObject(Sender: TObject): Boolean;
1675: Function IsEditing : Boolean
1676: Function ISEMPY : BOOLEAN
1677: Function IsEqual( Value : TParameters) : Boolean
1678: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1679: function IsEqualGUID(const guid1, guid2: TGUID): Boolean
1680: Function IsFirstNode : Boolean
1681: Function IsFloatZero( const X : Float) : Boolean
1682: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1683: Function IsFormOpen(const FormName: string): Boolean;
1684: Function IsFQDN( const S : String) : Boolean
1685: Function IsGrayScale : Boolean
1686: Function IsHex( AChar : Char) : Boolean;
1687: Function IsHexString(const AString: string): Boolean;
1688: Function IsHostname( const S : String) : Boolean
1689: Function IsInfinite( const AValue : Double) : Boolean
1690: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1691: Function IsInternet: boolean;
1692: Function IsLeadChar( ACh : Char) : Boolean
1693: Function IsLeapYear( Year : Word) : Boolean
1694: function IsLeapYear(Year: Word): Boolean)
1695: function IsLibrary: Boolean)
1696: Function ISLINE : BOOLEAN
1697: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1698: Function ISLINKEDTO( DATA SOURCE : TDATASOURCE) : BOOLEAN
1699: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1700: Function IsMatch( const Pattern, Text : string) : Boolean //Grep like RegEx
1701: Function IsMainAppWindow( Wnd : HWnd) : Boolean
1702: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1703: function IsMemoryManagerSet: Boolean)
1704: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1705: function IsMultiThread: Boolean)
1706: Function IsNumeric( AChar : Char) : Boolean;
1707: Function IsNumeric2( const AString : string) : Boolean;
1708: Function IsNTFS: Boolean;
1709: Function IsOctal( AChar : Char) : Boolean;
1710: Function IsOctalString(const AString: string): Boolean;
1711: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1712: function IsPathDelimiter(const S: string; Index: Integer): Boolean
1713: Function IsPM( const AValue : TDateTime) : Boolean
1714: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1715: Function IsPortAvailable( ComNum : Cardinal) : Boolean');
1716: Function IsComPortReal( ComNum : Cardinal) : Boolean');
1717: Function IsCOM( ComNum : Cardinal) : Boolean');
1718: Function IsCOMPort: Boolean');
1719: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1720: Function IsPrimerM( N : Cardinal) : Boolean //rabin miller
1721: Function IsPrimETD( N : Cardinal) : Boolean //trial division
1722: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1723: Function ISqrt( const I : Smallint) : Smallint
1724: Function IsReadOnly(const Filename: string): boolean;
1725: Function IsRectEmpty( const Rect : TRect) : Boolean
1726: function IsRectEmpty(const Rect: TRect): Boolean)
1727: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1728: Function ISRIGHTTOLEFT : BOOLEAN
1729: function IsRightToLeft: Boolean
1730: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1731: Function ISSEQUENCED : BOOLEAN
1732: Function IsSystemModule( const Module : HMODULE) : Boolean
1733: Function IsSystemResourcesMeterPresent : Boolean
1734: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1735: Function IsToday( const AValue : TdateTime) : Boolean
1736: function IsToday(const AValue: TDateTime): Boolean;
1737: Function IsTopDomain( const AStr : string) : Boolean
1738: Function IsUTF8LeadByte( Lead : Char) : Boolean
1739: Function IsUTF8String( const s : UTF8String) : Boolean
1740: Function IsUTF8TrailByte( Lead : Char) : Boolean
1741: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1742: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1743: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1744: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1745: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1746: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1747: Function IsValidIdent( Ident : string) : Boolean
1748: function IsValidIdent1(const Ident: string; AllowDots: Boolean): Boolean)
1749: Function IsValidIP( const S : String) : Boolean
1750: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean

```

```

1751: Function IsValidPNG(stream: TStream): Boolean;
1752: Function IsValidJPEG(stream: TStream): Boolean;
1753: Function IsValidISBN( const ISBN : AnsiString ) : Boolean
1754: Function IsVariantManagerSet: Boolean; //deprecated;
1755: Function IsVirtualPcGuest : Boolean;
1756: Function IsVmWareGuest : Boolean;
1757: Function IsVCLControl(Handle: HWnd): Boolean;
1758: Function IsWhiteString( const AStr : String ) : Boolean
1759: Function IsWindowResponding( Wnd : HWND; Timeout : Integer ) : Boolean
1760: Function IsWoW64: boolean;
1761: Function IsWin64: boolean;
1762: Function IsWow64String(var s: string): Boolean;
1763: Function IsWin64String(var s: string): Boolean;
1764: Function IsWindowsVista: boolean;
1765: Function isPowerof2(num: int64): boolean;
1766: Function powerOf2(exponent: integer): int64;
1767: function IsZero( const A: Extended; Epsilon: Extended): Boolean //overload;
1768: function IsZero1( const A: Double; Epsilon: Double): Boolean //overload;
1769: function IsZero2( const A: Single; Epsilon: Single): Boolean //overload;
1770: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1771: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1772: Function ItemRect( Index : Integer ) : TRect
1773: function ITEMRECT(INDEX:INTEGER):TRECT
1774: Function ItemWidth( Index : Integer ) : Integer
1775: Function JavahashCode(val: string): Integer;
1776: Function JosephusG(n,k: integer; var graphout: string): integer;
1777: Function JulianDateToDateTime( const AValue : Double ) : TDateTime
1778: Function JustName(PathName : string) : string; //in path and ext
1779: Function JvMessageBox( const Text, Caption : string; Flags : DWORD ) : Integer;
1780: Function JvMessageBox1( const Text : string; Flags : DWORD ) : Integer;
1781: Function KeepAlive : Boolean
1782: Function KeysToShiftState(Keys: Word): TShiftState;
1783: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1784: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1785: Function KeyboardStateToShiftState: TShiftState; overload;
1786: Function Languages : TLanguages
1787: Function Last : TClass
1788: Function Last : TComponent
1789: Function Last : TObject
1790: Function LastDelimiter( Delimiters, S : string ) : Integer
1791: function LastDelimiter(const Delimiters: string; const S: string): Integer
1792: Function LastPos( const ASubstr : string; const AStr : string ) : Integer
1793: Function Latitude2WGS84(lat: double): double;
1794: Function LCM(m,n:longint):longint;
1795: Function LCMJ( const X, Y : Cardinal ) : Cardinal
1796: Function Ldexp( const X : Extended; const P : Integer ) : Extended
1797: Function LeftStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
1798: Function LeftStr( const AText : WideString; const ACount : Integer ) : WideString;
1799: function Length: Integer;
1800: Procedure LetfileList(FileList: TStringlist; apath: string);
1801: function lengthmp3(mp3path: string):integer;
1802: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect ) : Boolean;
1803: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect ) : Boolean;
1804: function LinesCount(sfilename:string):extended;
1805: function TextFileLineCount(const FileName: string): integer;
1806: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1807: function LineStart(Buffer, BufPos: PChar): PChar
1808: function LineStart(Buffer, BufPos: PChar): PChar)
1809: function ListSeparator: char;
1810: function Ln(x: Extended): Extended;
1811: Function LnXPI( const X : Extended ) : Extended
1812: function Lo(vdat: word): byte;
1813: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1814: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean ) : Boolean
1815: Function LoadfileAsString( const FileName : string ) : string
1816: Function LoadFromFile( const FileName : string ) : TBitmapLoader
1817: function Loadfile(const FileName: TFileName): string;
1818: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1819: Function LoadPackage(const Name: string): HMODULE
1820: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle ) : HGLOBAL
1821: Function LoadStr( Ident : Integer ) : string
1822: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1823: Function LoadWideStr( Ident : Integer ) : WideString
1824: Function LOCATE( const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS ) : BOOLEAN
1825: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint ) : HResult
1826: Function LockServer( fLock : LongBool ) : HRESULT
1827: Function LockVolume( const Volume : string; var Handle : THandle ) : Boolean
1828: Function Log( const X : Extended ) : Extended
1829: Function Log10( const X : Extended ) : Extended
1830: Function Log2( const X : Extended ) : Extended
1831: function LogBase10(X: Float): Float;
1832: Function LogBase2(X: Float): Float;
1833: Function LogBaseN(Base, X: Float): Float;
1834: Function LogN( const Base, X : Extended ) : Extended
1835: Function LogOffOS : Boolean
1836: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string ) : Boolean
1837: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1838: Function LongDateFormat: string;

```

```

1839: function LongTimeFormat: string;
1840: Function LongWordToFourChar( ACARDINAL : LongWord) : string
1841: Function LOOKUP( const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1842: Function LookupName( const name : string) : TInAddr
1843: Function LookupService( const service : string) : Integer
1844: function Low: Int64;
1845: Function LowerCase( S : string) : string
1846: Function Lowercase(s : AnyString) : AnyString;
1847: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1848: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1849: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1850: function MainInstance: longword
1851: function MainThreadID: longword
1852: Function Map(x, in_min, in_max, out_min, out_max: integer): integer; //arduino
1853: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1854: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1855: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1856: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1857: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1858: Function MakeFile( const FileName: string): integer');
1859: function MakeLong(A, B: Word): Longint
1860: Function MakeTempFilename( const APath : String) : string
1861: Function MakeValidFileName( const Str : string) : string
1862: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1863: function MakeWord(A, B: Byte): Word
1864: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1865: Function MapDateTime( const DateFormatType:string; DateFormat:string; Value:string; ToCds:Boolean): string
1866: Function MapValues( Mapping : string; Value : string) : string
1867: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1868: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1869: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1870: Function MaskGetFldSeparator( const EditMask : string) : Integer
1871: Function MaskGetMaskBlank( const EditMask : string) : Char
1872: Function MaskGetMaskSave( const EditMask : string) : Boolean
1873: Function MaskIntLiteralToChar( IChar : Char) : Char
1874: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1875: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1876: Function MaskString( Mask, Value : String) : String
1877: Function Match( const sString : string) : TniRegularExpressionMatchResult
1878: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1879: Function Matches( const Filename : string) : Boolean
1880: Function MatchesMask( const Filename, Mask : string) : Boolean
1881: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1882: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1883: Function Max( AValueOne, AValueTwo : Integer) : Integer
1884: function Max(const x,y: Integer): Integer;
1885: Function Max1( const B1, B2 : Shortint) : Shortint;
1886: Function Max2( const B1, B2 : Smallint) : Smallint;
1887: Function Max3( const B1, B2 : Word) : Word;
1888: function Max3(const x,y,z: Integer): Integer;
1889: Function Max4( const B1, B2 : Integer) : Integer;
1890: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1891: Function Max6( const B1, B2 : Int64) : Int64;
1892: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1893: Function MaxFloat( const X, Y : Float) : Float
1894: Function MaxFloatArray( const B : TDynFloatArray) : Float
1895: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1896: function MaxIntValue(const Data: array of Integer):Integer
1897: Function MaxJ( const B1, B2 : Byte) : Byte;
1898: function MaxPath: string;
1899: function MaxValue(const Data: array of Double): Double)
1900: Function MaxCalc( const Formula : string) : Extended //math expression parser
1901: Procedure MaxCalcF( const Formula : string); //out to console memo2
1902: Function MaxCalcs( const Formula : string): string';
1903: function MD5(const fileName: string): string;
1904: Function Mean( const Data : array of Double) : Extended
1905: Function Median( const X : TDynFloatArray) : Float
1906: Function Memory: Pointer
1907: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1908: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1909: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1910: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1911: Function MessageDlg1(const
1912: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
1913: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
1914: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
1915: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
1916: Function MibToId( Mib : string) : string
1917: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1918: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1919: Function microsecondsToCentimeters(milliseconds: longint): longint; //340m/s speed of sound
1920: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1921: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1922: Procedure GetMidiOutputs( const List : TStrings)

```

```

1923: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
1924: Procedure GetGEOMap(C_form,apath: string; const Data: string); //C_form: [html/json/xml]
1925: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1926: Function MIDINoteToStr( Note : TMIDINote) : string
1927: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1928: Procedure GetMidiOutputs( const List : TStrings)
1929: Procedure MidiOutCheck( Code : MMResult)
1930: Procedure MidiInCheck( Code : MMResult)
1931: Function MillisecondOf( const AValue : TDateTime) : Word
1932: Function MillisecondOfTheDay( const AValue : TDateTime) : LongWord
1933: Function MillisecondOfTheHour( const AValue : TDateTime) : LongWord
1934: Function MillisecondOfTheMinute( const AValue : TDateTime) : LongWord
1935: Function MillisecondOfTheMonth( const AValue : TDateTime) : LongWord
1936: Function MillisecondOfTheSecond( const AValue : TDateTime) : Word
1937: Function MillisecondOfTheWeek( const AValue : TDateTime) : LongWord
1938: Function MillisecondOfTheYear( const AValue : TDateTime) : Int64
1939: Function MillisecondsBetween( const ANow, AThen : TDateTime) : Int64
1940: Function MillisecondSpan( const ANow, AThen : TDateTime) : Double
1941: Function milliToDateTIme( Millisecond : LongInt) : TDateTime');
1942: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1943: Function millis: int64;
1944: Function Min( AValueOne, AValueTwo : Integer) : Integer
1945: Function Min1( const B1, B2 : Shortint) : Shortint;
1946: Function Min2( const B1, B2 : Smallint) : Smallint;
1947: Function Min3( const B1, B2 : Word) : Word;
1948: Function Min4( const B1, B2 : Integer) : Integer;
1949: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1950: Function Min6( const B1, B2 : Int64) : Int64;
1951: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1952: Function MinClientRect : TRect;
1953: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1954: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1955: Function MinFloat( const X, Y : Float) : Float
1956: Function MinFloatArray( const B : TDynFloatArray) : Float
1957: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1958: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1959: Function MinimizeName( const FileName : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1960: function MinimizeName(const FileName: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1961: Function MinIntValue( const Data : array of Integer) : Integer
1962: function MinIntValue(const Data: array of Integer):Integer)
1963: Function MinJ( const B1, B2 : Byte) : Byte;
1964: Function MinuteOf( const AValue : TDateTime) : Word
1965: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1966: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1967: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1968: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1969: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1970: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1971: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1972: Function MinValue( const Data : array of Double) : Double
1973: function MinValue(const Data: array of Double): Double)
1974: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1975: Function MMCheck( const MciError : MCIERROr; const Msg : string) : MCIERROr
1976: Function ModFloat( const X, Y : Float) : Float
1977: Function ModifiedJulianDateToDateTIme( const AValue : Double) : TDateTime
1978: Function Modify( const Key : string; Value : Integer) : Boolean
1979: Function ModuleCacheID : Cardinal
1980: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1981: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1982: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1983: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1984: Function MonthOf( const AValue : TDateTime) : Word
1985: Function MonthOfTheYear( const AValue : TDateTime) : Word
1986: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1987: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1988: Function MonthStr( DateTIme : TDateTime) : string
1989: Function MouseCoord( X, Y : Integer) : TGridCoord
1990: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1991: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1992: Function MoveNext : Boolean
1993: Function MSecsToTimeStamp( MSecs : Comp) : TTImeStamp
1994: function MSecsToTimeStamp(MSecs: Comp): TTImeStamp)
1995: Function Name : string
1996: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
Double;PaymentTime:TPaymentTime):Extended
1997: function NetworkVolume(DriveChar: Char): string
1998: Function NEWBOTOMLINE : INTEGER
1999: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
2000: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORtCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK : TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
2001: Function NEWLINE : TMENUITEM
2002: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
2003: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
Right:PExprNode):PExprNode
2004: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
const ITEMS : array of TCMENUITEM) : TPOPUPMENU
2005: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
2006: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
TMenuItem;AENABLED:BOOL) : TMENUITEM

```

```

2007: Function NEWTOPLINE : INTEGER
2008: Function Next : TIdAuthWhatsNext
2009: Function NextCharIndex( S : String; Index : Integer) : Integer
2010: Function NextRecordSet : TCustomSQLDataSet
2011: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
2012: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
2013: Function NextToken : Char
2014: Function nextToken : WideString
2015: function NextToken:Char
2016: Function Norm( const Data : array of Double) : Extended
2017: Function NormalizeAngle( const Angle : Extended) : Extended
2018: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
2019: Function NormalizeRect( const Rect : TRect) : TRect
2020: function NormalizeRect(const Rect: TRect): TRect;
2021: Function Now : TDateTime
2022: function Now2: tDateTime
2023: Function NumProcessThreads : integer
2024: Function NumThreadCount : integer
2025: Function NthDayOfWeek( const AValue : TDateTime) : Word
2026: Function NtProductType : TNTProductType
2027: Function NtProductTypeString : string
2028: function Null: Variant;
2029: Function NullPoint : TPoint
2030: Function NullRect : TRect
2031: Function Null2Blank(aString:String):String;
2032: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue : Extended; PaymentTime : TPaymentTime) : Extended
2033: Function NumIP : integer
2034: function Odd(x: Longint): boolean;
2035: Function OffsetFromUTC : TDateTime
2036: Function OffsetPoint( const P, Offset : TPoint) : TPoint
2037: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
2038: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean
2039: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
2040: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
2041: Function OldBCDTocurr( const BCD : TBcd; var Curr : Currency) : Boolean
2042: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd;Precision:Int;Decimals:Integer):Boolean
2043: function OpenBit:Integer
2044: Function OpenDatabase : TDatabase
2045: Function OpenDatabase( const DatabaseName : string) : TDatabase
2046: Procedure OpenDir(adir: string);
2047: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
2048: Function OpenMap(const Data: string): boolean;
2049: Function OpenMapX(const Data: string): boolean;
2050: Function OpenObject( Value : PChar) : Boolean;
2051: Function OpenObject1( Value : string) : Boolean;
2052: Procedure OpenWeb('http://snippets.delphidabbler.com/#');
2053: Procedure OpenURL( const AText : TKkString');
2054: Procedure OpenWeb( const AText : TKkString');
2055: Procedure OpenBrowser( const AText : TKkString');
2056: Function OpenSession( const SessionName : string) : TSession
2057: Function OpenVolume( const Drive : Char) : THandle
2058: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
2059: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
2060: Function OrdToBinary( const Value : Byte) : string;
2061: Function OrdToBinary1( const Value : Shortint) : string;
2062: Function OrdToBinary2( const Value : Smallint) : string;
2063: Function OrdToBinary3( const Value : Word) : string;
2064: Function OrdToBinary4( const Value : Integer) : string;
2065: Function OrdToBinary5( const Value : Cardinal) : string;
2066: Function OrdToBinary6( const Value : Int64) : string;
2067: Function OSCheck( RetVal : Boolean) : Boolean
2068: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
2069: Function OSIdentToString( const OSIdent : DWORD) : string
2070: Function Output: Text)
2071: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
2072: Function Owner : TCustomListView
2073: function Owner : TPersistent
2074: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2075: Function PadL( pStr : String; plth : integer) : String
2076: Function PadL(s : AnyString;I : longInt) : AnyString;
2077: Function PadLCh( pStr : String; plth : integer; pChr : char) : String
2078: Function PadR( pStr : String; plth : integer) : String
2079: Function Padr(s : AnyString;I : longInt) : AnyString;
2080: Function PadRCh( pStr : String; plth : integer; pChr : char) : String
2081: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2082: Function Padz(s : AnyString;I : longInt) : AnyString;
2083: Function PaethPredictor( a, b, c : Byte) : Byte
2084: Function PARAMBYNAME( const VALUE : String) : TPARAM
2085: Function ParamByName( const Value : WideString) : TParameter
2086: Function ParamCount: Integer
2087: Function ParamsEncode( const ASrc : string) : string
2088: function ParamStr(Index: Integer): string
2089: Function ParseDate( const DateStr : string) : TDateTime
2090: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2091: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2092: Function PathAddExtension( const Path, Extension : string) : string
2093: Function PathAddSeparator( const Path : string) : string
2094: Function PathAppend( const Path, Append : string) : string

```

```

2095: Function PathBuildRoot( const Drive : Byte) : string
2096: Function PathCanonicalize( const Path : string) : string
2097: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2098: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2099: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2100: Function PathEncode( const ASrc : string) : string
2101: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2102: Function PathExtractFileNameNoExt( const Path : string) : string
2103: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2104: Function PathGetDepth( const Path : string) : Integer
2105: Function PathGetLongName( const Path : string) : string
2106: Function PathGetLongName2( Path : string) : string
2107: Function PathGetShortName( const Path : string) : string
2108: Function PathIsAbsolute( const Path : string) : Boolean
2109: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2110: Function PathIsDiskDevice( const Path : string) : Boolean
2111: Function PathIsUNC( const Path : string) : Boolean
2112: Function PathRemoveExtension( const Path : string) : string
2113: Function PathRemoveSeparator( const Path : string) : string
2114: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2115: Function Peek : Pointer
2116: Function Peek : TObject
2117: function PERFORM(MSG:CARDINAL;WPARAM:LPARAM;LONGINT):LONGINT
2118: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime) : Extended
2119: function Permutation(npr, k: integer): extended;
2120: function PermutationInt(npr, k: integer): Int64;
2121: Function PermutationJ( N, R : Cardinal) : Float
2122: Function Pi : Extended;
2123: Function PiE : Extended;
2124: Function PixelsToDialogsX( const Pixels : Word) : Word
2125: Function PixelsToDialogsY( const Pixels : Word) : Word
2126: Function PlaySound(s: pchar; flag, syncflag: integer): boolean;
2127: Function Point( X, Y : Integer) : TPoint
2128: function Point(X, Y: Integer): TPoint
2129: Function PointAssign( const X, Y : Integer) : TPoint
2130: Function PointDist( const P1, P2 : TPoint) : Double;
2131: function PointDist(const P1,P2: TFloatPoint): Double;
2132: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2133: function PointDist2(const P1,P2: TPoint): Double;
2134: Function PointEqual( const P1, P2 : TPoint) : Boolean
2135: Function PointIsNull( const P : TPoint) : Boolean
2136: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2137: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2138: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2139: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2140: Function Pop : Pointer
2141: Function Pop : TObject
2142: Function PopnStdDev( const Data : array of Double) : Extended
2143: Function PopnVariance( const Data : array of Double) : Extended
2144: Function PopulationVariance( const X : TDynFloatArray) : Float
2145: function Pos(SubStr, S: AnyString): Longint;
2146: Function PosEqual( const Rect : TRect) : Boolean
2147: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2148: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2149: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2150: Function Post1( AURL : string; const ASource : TStrings) : string;
2151: Function Post2( AURL : string; const ASource : TStream) : string;
2152: Function Post3( AURL : string; const ASource : TIIDMultiPartFormDataStream) : string;
2153: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2154: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2155: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2156: Function Power( const Base, Exponent : Extended) : Extended
2157: Function PowerBig(aval, n:integer): string;
2158: Function PowerIntJ( const X : Float; N : Integer) : Float;
2159: Function PowerJ( const Base, Exponent : Float) : Float;
2160: Function PowerOffOS : Boolean
2161: Function PreformatDateString( Ps : string) : string
2162: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2163: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2164: Function Printer : TPrinter
2165: Function ProcessPath2( const ABasePath:String; const APPath: String; const APathDelim:string): string
2166: Function ProcessResponse : TIidHTTPWhatsNext
2167: Function ProduceContent : string
2168: Function ProduceContentFromStream( Stream : TStream) : string
2169: Function ProduceContentFromString( const S : string) : string
2170: Function ProgIDToClassID(const ProgID: string): TGUID;
2171: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2172: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2173: Function PromptForFileName( var AFileName : string; const AFiler: string; const ADefaultExt : string;
const ATitle: string; const AInitialDir : string; SaveDialog: Boolean) : Boolean
2174: function PromptForFileName(var AFileName: string; const AFiler: string; const ADefaultExt: string;const
ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean
2175: Function PSScriptNeedFile(Sender:TObject;const OriginFileName:String;var FileName,Output:String):Boolean
2176: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2177: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2178: Function Push( AItem : Pointer) : Pointer

```

```

2179: Function Push( AObject : TObject ) : TObject
2180: Function Put1( AURL : string; const ASource : TStream ) : string;
2181: Function Pythagoras( const X, Y : Extended ) : Extended
2182: Function queryDLLInterface( var queryList : TStringList ) : TStringList
2183: Function queryDLLInterfaceTwo( var queryList : TStringList ) : TStringList
2184: Function QueryInterface( const IID: TGUID; out Obj ): HResult, CdStdCall
2185: Function queryPerformanceCounter2( mse: int64 ) : int64;
2186: //Function QueryPerformanceCounter( var lpPerformanceCount: Int64 ): LongBool; stdcall;
2187: //Function QueryPerformanceFrequency( mse: int64 ): boolean;
2188: Function QueryPerformanceCounter( var lcount: Int64 ): Boolean; stdcall;
2189: Function QueryPerformanceFrequency( var lfreq: int64 ): boolean; stdcall;
2190: Procedure QueryPerformanceCounter1( var aC: Int64 );
2191: Function QueryPerformanceFrequency1( var freq: int64 ): boolean;
2192: Function Quote( const ACommand : String ) : SmallInt
2193: Function QuotedStr( S : string ) : string
2194: Function RadToCycle( const Radians : Extended ) : Extended
2195: Function RadToDeg( const Radians : Extended ) : Extended
2196: Function RadToDeg( const Value : Extended ) : Extended;
2197: Function RadToDeg1( const Value : Double ) : Double;
2198: Function RadToDeg2( const Value : Single ) : Single;
2199: Function RadToGrad( const Radians : Extended ) : Extended
2200: Function RadToGrad( const Value : Extended ) : Extended;
2201: Function RadToGrad1( const Value : Double ) : Double;
2202: Function RadToGrad2( const Value : Single ) : Single;
2203: Function RandG( Mean, StdDev : Extended ) : Extended
2204: function Random( const ARange: Integer ): Integer;
2205: function random2(a: integer): double
2206: function RandomE: Extended;
2207: function RandomF: Extended;
2208: Function RandomFrom( const AValues : array of string ) : string;
2209: Function RandomRange( const AFrom, ATo : Integer ) : Integer
2210: function randSeed: longint
2211: Function RawToDataColumn( ACol : Integer ) : Integer
2212: Function Read : Char
2213: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint ) : HResult
2214: function Read(Buffer:String;Count:LongInt):LongInt
2215: Function ReadBinaryStream( const Section, Name : string; Value : TStream ) : Integer
2216: Function ReadBool( const Section, Ident : string; Default : Boolean ) : Boolean
2217: Function ReadCardinal( const AConvert : boolean ) : Cardinal
2218: Function ReadChar : Char
2219: Function ReadClient( var Buffer, Count : Integer ) : Integer
2220: Function ReadDate( const Section, Name : string; Default : TDateTime ) : TDateTime
2221: Function ReadDateTime( const Section, Name : string; Default : TDateTime ) : TDateTime
2222: Function ReadFloat( const Section, Name : string; Default : Double ) : Double
2223: Function ReadFromStack(const ARaiseExceptfDiscOn:Bool;ATimeout:Int;const ARaiseExceptTimeout:Bool):Int
2224: Function ReadInteger( const AConvert : boolean ) : Integer
2225: Function ReadInteger( const Section, Ident : string; Default : Longint ) : Longint
2226: Function ReadLn : string
2227: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer ) : string
2228: function ReadLn(question: string): string;
2229: Function readm: string; //read last line in memo2 - console!
2230: Function ReadLnWait( AFailCount : Integer ) : string
2231: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2232: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2233: Function ReadSmallInt( const AConvert : boolean ) : SmallInt
2234: Function ReadString( const ABytes : Integer ) : string
2235: Function ReadString( const Section, Ident, Default : string ) : string
2236: Function ReadString( Count : Integer ) : string
2237: Function ReadTime( const Section, Name : string; Default : TDateTime ) : TDateTime
2238: Function ReadTimeStampCounter : Int64
2239: Function RebootOS : Boolean
2240: Function Receive( ATimeOut : Integer ) : TReplyStatus
2241: Function ReceiveBuf( var Buf, Count : Integer ) : Integer
2242: Function ReceiveLength : Integer
2243: Function ReceiveText : string
2244: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2245: Function ReceiveSerialText: string
2246: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word ) : TDateTime
2247: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHour,AMin,ASec, AMilliSec:Word):TDateTime
2248: Function RecodeDay( const AValue : TDateTime; const ADay : Word ) : TDateTime
2249: Function RecodeHour( const AValue : TDateTime; const AHour : Word ) : TDateTime
2250: Function RecodeMillisecond( const AValue : TDateTime; const AMilliSecond : Word ) : TDateTime
2251: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word ) : TDateTime
2252: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word ) : TDateTime
2253: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word ) : TDateTime
2254: Function RecodeTime( const AValue : TDateTime; const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2255: Function RecodeYear( const AValue : TDateTime; const AYear : Word ) : TDateTime
2256: Function Reconcile( const Results : OleVariant) : Boolean
2257: Function Rect( Left, Top, Right, Bottom : Integer ) : TRect
2258: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect
2259: Function Rect2( const ATopLeft, ABottomRight : TPoint ) : TRect;
2260: Function RectAssign( const Left, Top, Right, Bottom : Integer ) : TRect
2261: Function RectAssignPoints( const TopLeft, BottomRight : TPoint ) : TRect
2262: Function RectBounds( const Left, Top, Width, Height : Integer ) : TRect
2263: Function RectCenter( const R : TRect ) : TPoint
2264: Function RectEqual( const R1, R2 : TRect ) : Boolean
2265: Function RectHeight( const R : TRect ) : Integer
2266: Function RectIncludesPoint( const R : TRect; const Pt : TPoint ) : Boolean

```

```

2267: Function RectIncludesRect( const R1, R2 : TRect ) : Boolean
2268: Function RectIntersection( const R1, R2 : TRect ) : TRect
2269: Function RectIntersectRect( const R1, R2 : TRect ) : Boolean
2270: Function RectIsEmpty( const R : TRect ) : Boolean
2271: Function RectIsNull( const R : TRect ) : Boolean
2272: Function RectIsSquare( const R : TRect ) : Boolean
2273: Function RectIsValid( const R : TRect ) : Boolean
2274: Function RectsAreValid( R : array of TRect ) : Boolean
2275: Function RectUnion( const R1, R2 : TRect ) : TRect
2276: Function RectWidth( const R : TRect ) : Integer
2277: Function RedComponent( const Color32 : TColor32 ) : Integer
2278: Function Refresh : Boolean
2279: Function RefStringlistCopy(aRefArray:TStringlist):TStringList;
2280: Function RegisterConversionFamily( const ADescription : string ) : TConvFamily
2281: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType ) : Boolean;
2282: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2283: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2284: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2285: Function ReleaseHandle : HBITMAP
2286: Function ReleaseHandle : HENHMETAFILE
2287: Function ReleaseHandle : HICON
2288: Function ReleasePalette : HPALETTE
2289: Function RemainderFloat( const X, Y : Float ) : Float
2290: Function Remove( AClass : TClass ) : Integer
2291: Function Remove( AComponent : TComponent ) : Integer
2292: Function Remove( AItem : Integer ) : Integer
2293: Function Remove( AItem : Pointer ) : Pointer
2294: Function Remove( AItem : TObject ) : TObject
2295: Function Remove( AObject : TObject ) : Integer
2296: Function RemoveBackslash( const PathName : string ) : string
2297: Function RemoveDF( aString : String ) : String //removes thousand separator
2298: Function RemoveDir( Dir : string ) : Boolean
2299: function RemoveDir(const Dir: string): Boolean
2300: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2301: Function RemoveFileExt( const FileName : string ) : string
2302: Function RemoveHeaderEntry( AHeader, AEntry : string ) : string
2303: Function RenameFile( OldName, NewName : string ) : Boolean
2304: function RenameFile(const OldName: string; const NewName: string): Boolean
2305: Function ReplaceStr( const AText, AFromText, AToText : string ) : string
2306: Function ReplaceText( const AText, AFromText, AToText : string ) : string
2307: Function Replicate(c : char;I : longInt) : String;
2308: Function Request : TWebRequest
2309: Function ResemblesText( const AText, AOther : string ) : Boolean
2310: Function Reset : Boolean
2311: function Reset2(mypath: string): TStringlist //string;
2312: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2313: Function ResourceLoad( Restype : TResType; const Name : string; MaskColor : TColor ) : Boolean
2314: Function Response : TWebResponse
2315: Function ResumeSupported : Boolean
2316: Function RETHINKHOTKEYS : BOOLEAN
2317: Function RETHINKLINES : BOOLEAN
2318: Function Retrieve( const MsgNum : Integer; AMsg : TIIdMessage ) : Boolean
2319: Function RetrieveCurrentDir : string
2320: Function RetrieveDeltas( const cdsArray : array of TClientDataset ) : Variant
2321: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIIdMessage ) : Boolean
2322: Function RetrieveMailBoxSize : integer
2323: Function RetrieveMsgSize( const MsgNum : Integer ) : Integer
2324: Function RetrieveProviders( const cdsArray : array of TClientDataset ) : Variant
2325: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings ) : boolean
2326: Function ReturnMIMEType( var MediaType, EncType : String ) : Boolean
2327: Function ReverseBits( Value : Byte) : Byte;
2328: Function ReverseBits1( Value : Shortint) : Shortint;
2329: Function ReverseBits2( Value : Smallint) : Smallint;
2330: Function ReverseBits3( Value : Word) : Word;
2331: Function ReverseBits4( Value : Cardinal) : Cardinal;
2332: Function ReverseBits4( Value : Integer) : Integer;
2333: Function ReverseBits5( Value : Int64) : Int64;
2334: Function ReverseBytes( Value : Word) : Word;
2335: Function ReverseBytes1( Value : Smallint) : Smallint;
2336: Function ReverseBytes2( Value : Integer) : Integer;
2337: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2338: Function ReverseBytes4( Value : Int64) : Int64;
2339: Function ReverseString( const AText : string ) : string
2340: Function ReversedDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var HName:Str):Bool;
2341: Function Revert : HRESULT
2342: Function RGB(R,G,B: Byte): TColor;
2343: Function RGB2BGR( const Color : TColor ) : TColor
2344: Function RGB2TColor( R, G, B : Byte) : TColor
2345: Function RGBToWebColorName( RGB : Integer ) : string
2346: Function RGBToWebColorStr( RGB : Integer ) : string
2347: Function RgbToHtml( Value : TColor ) : string
2348: Function HtmlToRgb(const Value: string): TColor;
2349: Function RightStr( const AStr : String; Len : Integer ) : String
2350: Function RightStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
2351: Function RightStr( const AText : WideString; const ACount : Integer ) : WideString;
2352: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2353: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2354: Function RotatePoint( Point : TFloaPoint; const Center : TFloaPoint; const Angle : Float) : TFloaPoint
2355: function RotatePoint(Point: TFloaPoint; const Center: TFloaPoint; const Angle: Double): TFloaPoint;

```

```

2356: Function Round(e : Extended) : Longint;
2357: Function Round64(e: extended): Int64;
2358: Function RoundAt( const Value : string; Position : SmallInt) : string
2359: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2360: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended) : Extended;');
2361: Function SimpleRoundTo(const AValue: Extended; const Abigit: TRoundToRange) : Extended;');
2362: Function RoundFrequency( const Frequency : Integer) : Integer
2363: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2364: Function RoundPoint( const X, Y : Double) : TPoint
2365: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2366: Function RowCount : Integer
2367: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2368: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2369: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2370: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2371: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2372: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2373: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2374: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2375: Function RunByteCode(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2376: Function RunCompiledScript2(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2377: Function S_AddBackSlash( const ADirName : string) : string
2378: Function S_AllTrim( const cStr : string) : string
2379: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2380: Function S_Cut( const cStr : string; const ilen : integer) : string
2381: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2382: Function S_DirExists( const ADir : string) : Boolean
2383: Function S_Empty( const cStr : string) : boolean
2384: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2385: Function S_LargeFontsActive : Boolean
2386: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2387: Function S_LTrim( const cStr : string) : string
2388: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2389: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2390: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2391: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2392: Function S_RTrim( const cStr : string) : string
2393: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2394: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2395: Function S_ShellExecute( afilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2396: Function S_Space( const iLen : integer) : String
2397: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2398: Function S_StrBlanksCuttoLong( const cstr : string; const ilen : integer) : string
2399: Function S_StrCRC32( const Text : string) : LongWORD
2400: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2401: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2402: Function S_StringtoUTF_8( const AString : string) : string
2403: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2404: function S_StrToReal(const cStr: string; var R: Double): Boolean
2405: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2406: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2407: Function S_UTF_8ToString( const AString : string) : string
2408: Function S_WBox( const AText : string) : integer
2409: Function SameDate( const A, B : TDateTime) : Boolean
2410: function SameDate(const A, B: TDateTime): Boolean;
2411: Function SameDateTime( const A, B : TDateTime) : Boolean
2412: function SameDateTime(const A, B: TDateTime): Boolean;
2413: Function SameFileName( S1, S2 : string) : Boolean
2414: Function SameText( S1, S2 : string) : Boolean
2415: function SameText(const S1: string; const S2: string): Boolean)
2416: Function SameTime( const A, B : TDateTime) : Boolean
2417: function Sametime(const A, B: TDateTime): Boolean;
2418: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2419: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2420: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2421: Function SampleVariance( const X : TDynFloatArray) : Float
2422: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2423: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2424: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2425: Function SaveToFile( const AFileName : TFileName) : Boolean
2426: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2427: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, afileName: string; openexcel: boolean): Boolean;
2428: Function ScanF(const aformat: String; const args: array of const): string;
2429: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2430: Function SearchBuf(Buf:PChar; BufLen:Integer; SelStart,SelLength:Integer;SearchString:String;Options: TStringSearchOptions):PChar
2431: Function SearchBuf2(Buf: string;SelStart,SelLength:Integer; SearchString: String;Options:TStringSearchOptions):Integer;
2432: function SearchRecattr: integer;
2433: function SearchRecExcludeAttr: integer;
2434: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2435: function SearchRecname: string;
2436: function SearchRecsize: integer;
2437: function SearchRecTime: integer;
2438: Function Sec( const X : Extended) : Extended
2439: Function Secant( const X : Extended) : Extended
2440: Function SecH( const X : Extended) : Extended
2441: Function SecondOf( const AValue : TDateTime) : Word
2442: Function SecondOfTheDay( const AValue : TDateTime) : LongWord

```

```

2443: Function SecondOfTheHour( const AValue : TDateTime) : Word
2444: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2445: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2446: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2447: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2448: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2449: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2450: Function SectionExists( const Section : string) : Boolean
2451: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2452: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2453: function Seek(Offset:LongInt;Origin:Word):LongInt
2454: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint) : Boolean;
2455: Function SelectDirectory( const Caption : string; const Root : WideString; var Directory : string;
    Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2456: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2457: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2458: Function SendBuf( var Buf, Count : Integer) : Integer
2459: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2460: Function SendCndl( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2461: Function SendKey( AppName : string; Key : Char) : Boolean
2462: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2463: Function SendStream( AStream : TStream) : Boolean
2464: Function SendStreamThenDrop( AStream : TStream) : Boolean
2465: Function SendText( const S : string) : Integer
2466: Function SendSerialData(Data: TByteArrayList; DataSize: cardinal): cardinal
2467: Function SendSerialText(Data: String): cardinal
2468: Function Sent : Boolean
2469: Function ServicesFilePath: string
2470: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2471: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2472: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2473: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2474: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2475: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2476: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2477: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2478: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2479: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2480: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2481: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2482: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2483: Function SetCurrentDir( Dir : string) : Boolean
2484: function SetCurrentDir(const Dir: string): Boolean
2485: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2486: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2487: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2488: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2489: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2490: Function SetEndOfFile(Handle: Integer) : LongBool; stdcall;
2491: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2492: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2493: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2494: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2495: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2496: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2497: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2498: Function SetLocalTime( Value : TDateTime) : boolean
2499: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2500: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2501: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2502: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2503: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2504: Function SetSize( libNewSize : Longint) : HResult
2505: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2506: Function Sgn( const X : Extended) : Integer
2507: function SHA1(const fileName: string): string;
2508: function SHA256(asr: string; amode: char): string)
2509: function SHA512(asr: string; amode: char): string)
2510: Function ShareMemoryManager : Boolean
2511: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2512: function Shelleexecute(hwnd : HWND; const FileName: string):integer; stdcall;
2513: Function ShellExecute3(afilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2514: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2515: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2516: function ShortDateFormat: string;
2517: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
    RTL:Boolean;EllipsisWidth:Int):WideString
2518: function ShortTimeFormat: string;
2519: function SHOWMODAL:INTEGER
2520: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
    TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult';
2521: Function
    ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2522: function ShowWindow(C1: HWND; C2: integer): boolean;
2523: procedure ShowMemory //in Dialog
2524: function ShowMemory2: string;
2525: Function ShutDownOS : Boolean
2526: Function Signe( const X, Y : Extended) : Extended
2527: Function Sign( const X : Extended) : Integer

```

```

2528: Function Sin(e : Extended) : Extended;
2529: Function sinc( const x : Double) : Double
2530: Function SinJ( X : Float) : Float
2531: Function Size( const AFileName : String) : Integer
2532: function SizeOf: Longint;
2533: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2534: function SlashSep(const Path, S: String): String
2535: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2536: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2537: Function SmallPoint(X, Y: Integer): TSmallPoint)
2538: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2539: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2540: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2541: Function SoundexProc( const AText, AOther : string) : Boolean
2542: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2543: Function SoundexWord( const AText : string) : Word
2544: Function SourcePos : Longint
2545: function SourcePos:Longint
2546: Function Split0( Str : string; const substr : string) : TStringList
2547: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2548: Function SQLRequiresParams( const SQL : WideString) : Boolean
2549: Function Sqr(e : Extended) : Extended;
2550: Function Sqrt(e : Extended) : Extended;
2551: Function StartIP : String
2552: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2553: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2554: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2555: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2556: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2557: Function StartOfAYear( const AYear : Word) : TDateTime
2558: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2559: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2560: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2561: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2562: Function StartsStr( const ASubText, AText : string) : Boolean
2563: Function StartsText( const ASubText, AText : string) : Boolean
2564: Function StartsWith( const ANSIstr, APattern : String) : Boolean
2565: Function StartsWith( const str : string; const sub : string) : Boolean
2566: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2567: Function StatusString( StatusCode : Integer) : string
2568: Function StdDev( const Data : array of Double) : Extended
2569: Function Stop : Float
2570: Function StopCount( var Counter : TJclCounter) : Float
2571: Function StoreColumns : Boolean
2572: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2573: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2574: Function StrAlloc( Size : Cardinal) : PChar
2575: function StrAlloc(Size: Cardinal): PChar)
2576: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2577: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2578: Function StrBufSize( Str : PChar) : Cardinal
2579: function StrBufSize(const Str: PChar): Cardinal)
2580: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2581: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2582: Function StrCat( Dest : PChar; Source : PChar) : PChar
2583: function StrCat(Dest: PChar; const Source: PChar): PChar)
2584: Function StrCharLength( Str : PChar) : Integer
2585: Function StrComp( Str1, Str2 : PChar) : Integer
2586: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2587: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2588: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2589: Function Stream_to_AnsiString( Source : TStream) : ansistring
2590: Function Stream_to_Base64( Source : TStream) : ansistring
2591: Function Stream_to_decimalbytes( Source : TStream) : string
2592: Function Stream2WideString( ostream : TStream) : WideString
2593: Function StreamtoAnsiString( Source : TStream) : ansistring
2594: Function StreamToByte( Source : TStream) : string
2595: Function StreamToDecimalbytes( Source : TStream) : string
2596: Function StreamtoOrd( Source : TStream) : string
2597: Function StreamToString( Source : TStream) : string
2598: Function StreamToString2( Source : TStream) : string
2599: Function StreamToString3( Source : TStream) : string
2600: Function StreamToString4( Source : TStream) : string
2601: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2602: Function StrEmpty( const sString : string) : boolean
2603: Function StrEnd( Str : PChar) : PChar
2604: function StrEnd(const Str: PChar): PChar)
2605: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2606: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2607: Function StrGet(var S : String; I : Integer) : Char;
2608: Function StrGet2(S : String; I : Integer) : Char;
2609: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2610: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2611: Function StrHtmlDecode( const AStr : String) : String
2612: Function StrHtmlEncode( const AStr : String) : String
2613: Function StrToBytes(const Value: String): TBytes;
2614: Function StrIComp( Str1, Str2 : PChar) : Integer
2615: Function StringOfChar(c : char;I : longInt) : String;
2616: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;

```

```

2617: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2618: Function StringRefCount(const s: String): integer;
2619: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2620: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2621: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2622: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2623: Function String.ToBoolean( const Ps : string) : Boolean
2624: function StringToColor(const S: string): TColor
2625: function StringToCursor(const S: string): TCursor;
2626: function StringToGUID(const S: string): TGUID)
2627: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2628: Function TStringToArray( const str : string; const delim : string) : TStringDynArray
2629: Function StringWidth( S : string) : Integer
2630: Function StrInternetToDateTime( Value : string) : TDateTime
2631: Function StrIsDateTime( const Ps : string) : Boolean
2632: Function StrIsFloatMoney( const Ps : string) : Boolean
2633: Function StrIsInteger( const S : string) : Boolean
2634: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2635: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2636: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2637: Function StrLen( Str : PChar) : Cardinal
2638: function StrLen(const Str: PChar): Cardinal)
2639: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2640: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2641: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2642: Function StrLower( Str : PChar) : PChar
2643: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2644: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2645: Function StrNew( Str : PChar) : PChar
2646: function StrNew(const Str: PChar): PChar)
2647: Function StrNextChar( Str : PChar) : PChar
2648: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2649: Function StrParse( var sString : string; const sDelimiters : string) : string;
2650: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2651: Function StrPas( Str : PChar) : string
2652: function StrPas(const Str: PChar): string)
2653: Function StrPCopy( Dest : PChar; Source : string) : PChar
2654: function StrPCopy(Dest: PChar; const Source: string): PChar)
2655: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2656: Function StrPos( Str1, Str2 : PChar) : PChar
2657: Function StrScan(const Str: PChar; Chr: Char): PChar)
2658: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2659: Function StrToBcd( const AValue : string) : TBcd
2660: Function StrToBool( S : string) : Boolean
2661: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2662: Function StrToCard( const AStr : String) : Cardinal
2663: Function StrToConv( AText : string; out AType : TConvType) : Double
2664: Function StrToCurr( S : string) : Currency;
2665: function StrToCurr(const S: string): Currency)
2666: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2667: Function StrToDate( S : string) : TDateTime;
2668: function StrToDate(const s: string): TDateTime;
2669: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2670: Function StrToDatetime( S : string) : TDateTime;
2671: function StrToDateTime(const S: string): TDateTime)
2672: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2673: Function StrToDay( const ADay : string) : Byte
2674: Function StrToFloat( S : string) : Extended;
2675: function StrToFloat(s: String): Extended;
2676: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2677: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2678: Function StrToFloat( S : string) : Extended;
2679: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2680: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2681: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2682: Function StrToCurr( S : string) : Currency;
2683: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2684: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2685: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2686: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2687: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2688: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2689: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2690: Function StrToDateTime( S : string; Tdate: TDateTime);
2691: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2692: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2693: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2694: Function StrToInt( S : string) : Integer
2695: function StrToInt(s: String): Longint;
2696: Function StrToInt64( S : string) : Int64
2697: function StrToInt64(s: String): int64;
2698: Function StrToInt64Def( S : string; Default : Int64) : Int64
2699: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2700: Function StrToIntDef( S : string; Default : Integer) : Integer
2701: function StrToIntDef(const S: string; Default: Integer): Integer)
2702: function StrToIntDef(s: String; def: Longint): Longint;
2703: Function StrToMonth( const AMonth : string) : Byte
2704: Function StrToTime( S : string) : TDateTime;
2705: function StrToTime(const S: string): TDateTime)

```

```

2706: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2707: Function StrToWord( const Value : String) : Word;
2708: Function StrToXmlDate( const DateStr : string; const Format : string) : string;
2709: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string;
2710: Function StrToXmlTime( const TimeStr : string; const Format : string) : string;
2711: Function StrUpper( Str : PChar) : PChar;
2712: Function StuffString( const AText : string; AStart, ALen : Cardinal; const ASubText : string) : string;
2713: Function Sum( const Data : array of Double) : Extended;
2714: Function SumFloatArray( const B : TDynFloatArray) : Float;
2715: Function SumInt( const Data : array of Integer) : Integer;
2716: Function SumOfSquares( const Data : array of Double) : Extended;
2717: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float;
2718: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float;
2719: Function SumSquareFloatArray( const B : TDynFloatArray) : Float;
2720: Function Supports( CursorOptions : TCursorOptions) : Boolean;
2721: Function SupportsClipboardFormat( AFormat : Word) : Boolean;
2722: Function SwapWord(w : word) : word;
2723: Function SwapInt(i : integer) : integer;
2724: Function SwapLong(L : longint) : longint;
2725: Function Swap(i : integer) : integer;
2726: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended;
2727: Function SyncTime : Boolean;
2728: Function SysErrorMessage( ErrorCode : Integer) : string;
2729: function SysErrorMessage(ErrorCode: Integer): string;
2730: Function SystemTimeToDate( SystemTime : TSystemTime) : TDateTime;
2731: function SystemTimeToDate( const SystemTime : TSystemTime) : TDateTime;
2732: Function SysStringLen(const S: WideString) : Integer; stdcall;
2733: Function TabRect( Index : Integer) : TRect;
2734: Function Tan( const X : Extended) : Extended;
2735: Function TaskMessageDlg(const Title,
  Msg:string; DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2736: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2737: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx : Longint; X, Y : Integer) : Integer;
2738: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2739: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2740: Function TaskMessageDlgPosHelp1( const Title, Msg:string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn) : Integer;
2741: Function TenToY( const Y : Float) : Float;
2742: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult;
2743: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult;
2744: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2745: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2746: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2747: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2748: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2749: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2750: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2751: Function TestBits( const Value, Mask : Byte) : Boolean;
2752: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2753: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2754: Function TestBits3( const Value, Mask : Word) : Boolean;
2755: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2756: Function TestBits5( const Value, Mask : Integer) : Boolean;
2757: Function TestBits6( const Value, Mask : Int64) : Boolean;
2758: Function TestFDIVInstruction : Boolean;
2759: function TestStreamFormat(Stream: TStream) : TStreamOriginalFormat;
2760: Function TextExtent( const Text : string) : TSize;
2761: function TextHeight(Text: string) : Integer;
2762: Function TextIsSame( const A1 : string; const A2 : string) : Boolean;
2763: Function TextStartsWith( const S, SubS : string) : Boolean;
2764: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue) : Boolean;
2765: Function ConvInteger(i : integer):string;
2766: Function IntegerToText(i : integer):string;
2767: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT;
2768: function TextWidth(Text: string) : Integer;
2769: Function ThreadCount : integer;
2770: function ThousandSeparator: char;
2771: Function Ticks : Cardinal;
2772: Function Time : TDateTime;
2773: function Time: TDateTime;
2774: function TimeGetTime: int64;
2775: Function TimeOf( const AValue : TDateTime) : TDateTime;
2776: function TimeSeparator: char;
2777: function TimeStampToDate( const TimeStamp: TTimeStamp) : TDateTime;
2778: Function TimeStampToMSECS( TimeStamp : TTimeStamp) : Comp;
2779: function TimeStampToMSECS( const TimeStamp: TTimeStamp) : Comp;
2780: Function TimeToStr( DateTime : TDateTime) : string;
2781: function TimeToStr( const DateTime: TDateTime) : string;
2782: Function TimeZoneBias : TDateTime;
2783: Function ToCommon( const AValue : Double) : Double;
2784: function ToCommon( const AValue: Double): Double;
2785: Function Today : TDateTime;
2786: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2787: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2788: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;

```

```

2789: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2790: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2791: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2792: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2793: function TokenComponentIdent:String
2794: Function TokenFloat : Extended
2795: function TokenFloat:Extended
2796: Function TokenInt : Longint
2797: function TokenInt:LongInt
2798: Function TokenString : string
2799: function TokenString:String
2800: Function TokenSymbolIs( const S : string) : Boolean
2801: function TokenSymbolIs(S:String):Boolean
2802: Function Tomorrow : TDateTime
2803: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2804: Function ToString : string
2805: Function TotalVariance( const Data : array of Double) : Extended
2806: Function Trace2( AURL : string) : string;
2807: Function TrackMenu( Button : TToolbutton) : Boolean
2808: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2809: Function TranslateURI( const URI : string) : string
2810: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2811: Function TransparentStretchBlt( DstDC : HDC; DstX, DstY, DstW, DstH: Integer; SrcDC:HDC;SrcX,SrcY,SrcW, SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2812: Function Trim( S : string) : string;
2813: Function Blank( S : string) : string; //alias to Trim
2814: Function Trim( S : WideString) : WideString;
2815: Function Trim(s : AnyString) : AnyString;
2816: Function TrimAllOf( ATrim, AText : String) : String
2817: Function TrimLeft( S : string) : string;
2818: Function TrimLeft( S : WideString) : WideString;
2819: function TrimLeft(const S: string): string
2820: Function TrimRight( S : string) : string;
2821: Function TrimRight( S : WideString) : WideString;
2822: function TrimRight(const S: string): string
2823: function TrueBoolStrs: array of string
2824: Function Trunc(e : Extended) : Longint;
2825: Function Trunc64(e: extended): Int64;
2826: Function TruncPower( const Base, Exponent : Float) : Float
2827: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2828: Function TryConvTypeToFamilyL( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2829: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2830: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2831: Function TryEncodeDateMonthWeek( const AY, AMonth, AWeekOfMonth, ADayOfWeek:Word;var AValue:TDateTime): Boolean
2832: Function TryEncodeDateTime( const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out AValue:TDateTime):Boolean
2833: Function TryEncodeDateWeek( const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2834: Function TryEncodeDayOfWeekInMonth( const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out AVal:TDateTime):Bool
2835: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2836: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2837: Function TryJulianToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2838: Function TryLock : Boolean
2839: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2840: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecound, AMilliSecond : Word; out AResult : TDateTime) : Boolean
2841: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2842: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2843: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2844: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2845: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2846: Function TryStrToInt(const S: Ansistring; var I: Integer): Boolean;
2847: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2848: function TryStrToBool(const S: string; out Value: Boolean): Boolean;
2849: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2850: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2851: Function TwoToY( const Y : Float) : Float
2852: Function UCS4StringToWideString( const S : UCS4String) : WideString
2853: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2854: function Unassigned: Variant;
2855: Function UndoLastChange( FollowChange : Boolean) : Boolean
2856: function UniCodeToStr(Value: string): string;
2857: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2858: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean
2859: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2860: Function UnixPathToDosPath( const Path : string) : string
2861: Function UnixToDateTIme( const AValue : Int64) : TDateTime
2862: function UnixToDateTIme(U: Int64): TDateTime;
2863: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2864: Function UnlockResource( ResData : HGLOBAL) : LongBool
2865: Function UnlockVolume( var Handle : THandle) : Boolean
2866: Function UnMaskString( Mask, Value : String) : String
2867: function UpCase(ch : Char) : Char;
2868: Function UpCaseFirst( const AStr : string) : string
2869: Function UpCaseFirstWord( const AStr : string) : string
2870: Function UpdateAction( Action : TBasicAction) : Boolean
2871: Function UpdateKind : TUpdateKind
2872: Function UPDATESTATUS : TUPDATESTATUS
2873: Function UpperCase( S : string) : string

```

```

2874: Function Uppercase(s : AnyString) : AnyString;
2875: Function URLDecode( ASrc : string) : string
2876: Function URLEncode( const ASrc : string) : string
2877: Function UseRightToLeftAlignment : Boolean
2878: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2879: Function UseRightToLeftReading : Boolean
2880: Function UTF8CharLength( Lead : Char) : Integer
2881: Function UTF8CharSize( Lead : Char) : Integer
2882: Function UTF8Decode( const S : UTF8String) : WideString
2883: Function UTF8Encode( const WS : WideString) : UTF8String
2884: Function UTF8LowerCase( const S : UTF8String) : UTF8String
2885: Function Utf8ToAnsi( const S : UTF8String) : string
2886: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2887: Function UTF8UpperCase( const S : UTF8String) : UTF8String
2888: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2889: Function ValidParentForm(control: TControl): TForm
2890: Function Value : Variant
2891: Function ValueExists( const Section, Ident : string) : Boolean
2892: Function ValueOf( const Key : string) : Integer
2893: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2894: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2895: Function VarArrayFromStrings( Strings : TStrings) : Variant
2896: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2897: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2898: Function VarFMTBcd : TVarType
2899: Function VarFMTBcdCreate1 : Variant;
2900: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2901: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2902: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2903: Function Variance( const Data : array of Double) : Extended
2904: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2905: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2906: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2907: Function VariantGetElement( const V : Variant; I1 : integer) : Variant;
2908: Function VariantGetElement1( const V : Variant; I1, I2 : integer) : Variant;
2909: Function VariantGetElement2( const V : Variant; I1, I2, I3 : integer) : Variant;
2910: Function VariantGetElement3( const V : Variant; I1, I2, I3, I4 : integer) : Variant;
2911: Function VariantGetElement4( const V : Variant; I1, I2, I3, I4, I5 : integer) : Variant;
2912: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2913: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2914: Function VariantNeg( const V1 : Variant) : Variant
2915: Function VariantNot( const V1 : Variant) : Variant
2916: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2917: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2918: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2919: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2920: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2921: function VarIsEmpty(const V: Variant): Boolean;
2922: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2923: function VarIsNull(const V: Variant): Boolean;
2924: Function VarToBcd( const AValue : Variant) : TBcd
2925: function VarType(const V: Variant): TVarType;
2926: Function VarType( const V : Variant) : TVarType
2927: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2928: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2929: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2930: Function VarIsByRef( const V : Variant) : Boolean
2931: Function VarIsEmpty( const V : Variant) : Boolean
2932: Procedure VarCheckEmpty( const V : Variant)
2933: Function VarIsNull( const V : Variant) : Boolean
2934: Function VarIsClear( const V : Variant) : Boolean
2935: Function VarIsCustom( const V : Variant) : Boolean
2936: Function VarIsOrdinal( const V : Variant) : Boolean
2937: Function VarIsFloat( const V : Variant) : Boolean
2938: Function VarIsNumeric( const V : Variant) : Boolean
2939: Function VarIsStr( const V : Variant) : Boolean
2940: Function VarToStr( const V : Variant) : string
2941: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2942: Function VarToWideStr( const V : Variant) : WideString
2943: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2944: Function VarToDateTime( const V : Variant) : TDateTime
2945: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2946: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2947: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2948: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThanOrEqual, vrNotEqual )'
2949: Function VarSameValue( const A, B : Variant) : Boolean
2950: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2951: Function VarIsEmptyParam( const V : Variant) : Boolean
2952: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2953: Function VarIsError1( const V : Variant) : Boolean;
2954: Function VarAsError( AResult : HRESULT) : Variant
2955: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2956: Function VarIsArray( const A : Variant) : Boolean;
2957: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2958: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2959: Function VarArrayOf( const Values : array of Variant) : Variant
2960: Function VarArrayRef( const A : Variant) : Variant
2961: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2962: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean

```

```

2963: Function VarArrayDimCount( const A : Variant) : Integer
2964: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2965: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2966: Function VarArrayLock( const A : Variant) : __Pointer
2967: Procedure VarArrayUnlock( const A : Variant)
2968: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2969: Procedure VarArrayPut( var A: Variant; const Value : Variant; const Indices : array of Integer)
2970: Procedure DynArrayToVariant( var V : Variant; const DynArray : __Pointer; TypeInfo : __Pointer)
2971: Procedure DynArrayFromVariant( var DynArray : __Pointer; const V : Variant; TypeInfo : __Pointer)
2972: Function Unassigned : Variant
2973: Function Null : Variant
2974: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2975: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2976: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2977: function VectorDot(const V1,V2: TFloatPoint): Double;
2978: Function VectorLengthSqr( const V : TFloatPoint) : Double
2979: function VectorLengthSqr(const V: TFloatPoint): Double;
2980: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2981: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2982: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2983: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2984: Function Verify( AUserName : String) : String
2985: Function Versine( X : Float) : Float
2986: function VersionCheck: boolean;
2987: function VersionCheckAct: string;
2988: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2989: Function VersionLanguageName( const LangId : Word) : string
2990: Function VersionResourceAvailable( const FileName : string) : Boolean
2991: Function Visible : Boolean
2992: function VolumeID(DriveChar: Char): string
2993: Function WaitFor( const AString : string) : string
2994: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2995: Function WaitFor1 : TWaitResult;
2996: Function WaitForData( Timeout : Longint) : Boolean
2997: Function WebColorNameToColor( WebColorName : string) : TColor
2998: Function WebColorStrToColor( WebColor : string) : TColor
2999: Function WebColorToRGB( WebColor : Integer) : Integer
3000: Function wGet(aURL, afile: string): boolean;
3001: Function wGet2(aURL, afile: string): boolean; //without file open
3002: Function wGetX(aURL, afile: string): boolean;
3003: Function wGetX2(aURL, afile: string): boolean; //without file open
3004: Function WebGet(aURL, afile: string): boolean;
3005: Function WebExists: boolean; //alias to isinternet
3006: Function WeekOf( const AValue : TDateTime) : Word
3007: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
3008: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
3009: Function WeekOfTheYear( const AValue : TDateTime) : Word;
3010: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
3011: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
3012: Function WeeksInAYear( const AYear : Word) : Word
3013: Function WeeksInYear( const AValue : TDateTime) : Word
3014: Function WeekSpan( const ANow, ATThen : TDateTime) : Double
3015: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineStyle) : WideString
3016: Function WideCat( const x, y : WideString) : WideString
3017: Function WideCompareStr( S1, S2 : WideString) : Integer
3018: function WideCompareStr(const S1: WideString; const S2: WideString): Integer
3019: Function WideCompareText( S1, S2 : WideString) : Integer
3020: function WideCompareText(const S1: WideString; const S2: WideString): Integer
3021: Function WideCopy( const src : WideString; index, count : Integer) : WideString
3022: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
3023: Function WideEqual( const x, y : WideString) : Boolean
3024: function WideFormat(const Format: WideString; const Args: array of const): WideString)
3025: Function WideGreater( const x, y : WideString) : Boolean
3026: Function WideLength( const src : WideString) : Integer
3027: Function WideLess( const x, y : WideString) : Boolean
3028: Function WideLowerCase( S : WideString) : WideString
3029: function WideLowerCase(const S: WideString): WideString)
3030: Function WidePos( const src, sub : WideString) : Integer
3031: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
3032: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
3033: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
3034: Function WideSameStr( S1, S2 : WideString) : Boolean
3035: function WideSameStr(const S1: WideString; const S2: WideString): Boolean
3036: Function WideSameText( S1, S2 : WideString) : Boolean
3037: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
3038: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
3039: Function WideStringToUCS4String( const S : WideString) : UCS4String
3040: Function WideUpperCase( S : WideString) : WideString
3041: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
3042: function Win32Check(RetVal: boolean): boolean)
3043: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
3044: Function Win32RestoreFile( const FileName : string) : Boolean
3045: Function Win32Type : TIWin32Type
3046: Function WinColor( const Color32 : TColor32) : TColor
3047: function winexec(FileName: pchar; showCmd: integer): integer;
3048: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3049: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3050: Function WithinPastDays( const ANow, ATThen : TDateTime; const ADays : Integer) : Boolean
3051: Function WithinPastHours( const ANow, ATThen : TDateTime; const AHours : Int64) : Boolean

```

```

3052: Function WithinPastMilliseconds( const ANow, AThen : TDateTime; const AMilliseconds : Int64) : Boolean
3053: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
3054: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
3055: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASconds : Int64) : Boolean
3056: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
3057: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
3058: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
3059: Function WordToStr( const Value : Word) : String
3060: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
3061: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
3062: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
3063: Function WorkArea : Integer
3064: Function WrapText( Line : string; MaxCol : Integer) : string;
3065: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
3066: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3067: function Write(Buffer:String;Count:LongInt):LongInt
3068: Function WriteClient( var Buffer, Count : Integer) : Integer
3069: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
3070: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
3071: Function WriteString( const AString : string) : Boolean
3072: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3073: Function wvprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3074: Function wsprintf( Output : PChar; Format : PChar) : Integer
3075: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3076: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3077: Function XorDecode( const Key, Source : string) : string
3078: Function XorEncode( const Key, Source : string) : string
3079: Function XorString( const Key, Src : ShortString) : ShortString
3080: Function Yield : Bool
3081: Function YearOf( const AValue : TDateTime) : Word
3082: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3083: Function YearSpan( const ANow, AThen : TDateTime) : Double
3084: Function Yesterday : TDateTime
3085: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3086: Function( const Name : string; Proc : TUserFunction)
3087: Function using Special_Scholz from 3.8.5.0
3088: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3089: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3090: Function FloatToTime2Dec(value:Extended):Extended;
3091: Function MinToStd(value:Extended):Extended;
3092: Function MinToStdAsString(value:Extended):String;
3093: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3094: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3095: Function Round2Dec (zahl:Extended):Extended;
3096: Function GetAngle(x,y:Extended):Double;
3097: Function AddAngle(a1,a2:Double):Double;
3098:
3099: ****
3100: unit uPSI_StText;
3101: ****
3102: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3103: Function TextFileSize( var F : TextFile) : LongInt
3104: Function TextPos( var F : TextFile) : LongInt
3105: Function TextFlush( var F : TextFile) : Boolean
3106:
3107: ****
3108: from JvVCLUtils;
3109: ****
3110: { Windows resources (bitmaps and icons) VCL-oriented routines }
3111: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3112: procedure DrawBitmapRectTransparent(Dest: TCNvas; DstX, DstY: Integer; SrcRect: TRect,
  DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3113: procedure StretchBitmapRectTransparent(Dest: TCNvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
  Bitmap: TBitmap; TransparentColor: TColor);
3114: function MakeBitmap(ResID: PChar): TBitmap;
3115: function MakeBitmapID(ResID: Word): TBitmap;
3116: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3117: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3118: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3119: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
  HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3120: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3121: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3122: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3123: {$IFDEF WIN32}
3124: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCNva;
  X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3125: {$ENDIF}
3126: function MakeIcon(ResID: PChar): TIcon;
3127: function MakeIconID(ResID: Word): TIcon;
3128: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3129: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3130: {$IFDEF WIN32}
3131: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3132: {$ENDIF}
3133: { Service routines }
3134: procedure NotImplemented;
3135: procedure ResourceNotFound(ResID: PChar);
3136: function PointInRect(const P: TPoint; const R: TRect): Boolean;

```

```

3139: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3140: function PaletteColor(Color: TColor): Longint;
3141: function WidthOf(R: TRect): Integer;
3142: function HeightOf(R: TRect): Integer;
3143: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3144: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3145: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3146: procedure Delay(MSecs: Longint);
3147: procedure DeleteLine(StrList: TStringList; SearchPattern: String);
3148: procedure CenterControl(Control: TControl);
3149: Function PaletteEntries( Palette : HPALETTE) : Integer
3150: Function WindowClassName( Wnd : HWND) : string
3151: Function ScreenWorkArea : TRect
3152: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3153: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3154: Procedure ActivateWindow( Wnd : HWND)
3155: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3156: Procedure CenterWindow( Wnd : HWND)
3157: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3158: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3159: Function DialogsToPixelsX( Dlgs : Word) : Word
3160: Function DialogsToPixelsY( Dlgs : Word) : Word
3161: Function PixelsToDialogsX( Pixs : Word) : Word
3162: Function PixelsToDialogsY( Pixs : Word) : Word
3163: {$IFDEF WIN32}
3164: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3165: function MakeVariant(const Values: array of Variant): Variant;
3166: {$ENDIF}
3167: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3168: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3169: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3170: {$IFDEF CBUILER}
3171: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3172: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3173: {$ELSE}
3174: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3175: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3176: {$ENDIF CBUILER}
3177: function IsForegroundTask: Boolean;
3178: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3179: function GetAveCharSize(Canvas: TCanvas): TPoint;
3180: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3181: procedure FreeUnusedOle;
3182: procedure Beep;
3183: function GetWindowsVersionJ: string;
3184: function LoadDLL(const LibName: string): THandle;
3185: function RegisterServer(const ModuleName: string): Boolean;
3186: {$IFNDEF WIN32}
3187: function IsLibrary: Boolean;
3188: {$ENDIF}
3189: { Gradient filling routine }
3190: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3191: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction: TFillDirection; Colors: Byte);
3192: { String routines }
3193: function GetEnvVar(const VarName: string): string;
3194: function AnsiUpperFirstChar(const S: string): string;
3195: function StringToPChar(var S: string): PChar;
3196: function StrPAalloc(const S: string): PChar;
3197: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
3198: function DropT(const S: string): string;
3199: { Memory routines }
3200: function AllocMemo(Size: Longint): Pointer;
3201: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3202: procedure FreeMemo(var fpBlock: Pointer);
3203: function GetMemosize(fpBlock: Pointer): Longint;
3204: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3205: {$IFDEF COMPILERS_UP}
3206: procedure FreeAndNil(var Obj);
3207: {$ENDIF}
3208: // from PNGLoader
3209: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3210: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3211: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3212: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3213: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style : TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect //TButtons
3214: Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3215: Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3216: AddConstantN('CTL3D_ALL','LongWord').SetUInt($FFFF);
3217: //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3218: //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3219: //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3220: Procedure SetIMEMode( hWnd : HWND; Mode : TIMEMode)
3221: Procedure SetIMEName( Name : TIMEName)
3222: Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3223: Function Imm32GetContext( hWnd : HWND) : HIMC
3224: Function Imm32ReleaseContext( hWnd : HWND; hIMC : HIMC) : Boolean
3225: Function Imm32GetConversionStatus( hIMC : HIMC; var Conversion, Sentence : longword) : Boolean

```

```

3226: Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3227: Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3228: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3229: //Function Imm32SetCompositionFont( hImc : HIMC; lpLogFont : PLOGFONTA) : Boolean
3230: Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3231: Function Imm32IsIME( hKl : longword) : Boolean
3232: Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3233: Procedure DragDone( Drop : Boolean)
3234:
3235:
3236: //*****added from jvjcutilis
3237: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3238: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3239: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3240: function IsPositiveResult(Value: TModalResult): Boolean;
3241: function IsNegativeResult(Value: TModalResult): Boolean;
3242: function IsAbortResult(const Value: TModalResult): Boolean;
3243: function StripAllFromResult(const Value: TModalResult): TModalResult;
3244: // returns either BrightColor or DarkColor depending on the luminance of AColor
3245: // This function gives the same result (AFAIK) as the function used in Windows to
3246: // calculate the desktop icon text color based on the desktop background color
3247: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3248: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3249:
3250: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3251:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3252:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3253:   var LinkName: string; Scale: Integer = 100); overload;
3254: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3255:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3256:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3257:   var LinkName: string; Scale: Integer = 100); overload;
3258: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3259:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3260: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3261:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3262:   Scale: Integer = 100): string;
3263: function HTMLPlainText(const Text: string): string;
3264: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3265:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3266: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3267:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3268: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3269: function HTMLPrepareText(const Text: string): string;
3270:
3271: ***** uPSI_JvAppUtils;
3272: Function GetDefaultSection( Component : TComponent) : string
3273: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3274: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3275: Function GetDefaultIniName : string
3276: //OnGetDefaultIniName','OnGetDefaultIniName';
3277: Function GetDefaultIniRegKey : string
3278: Function FindForm( FormClass : TFormClass) : TForm
3279: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3280: Function ShowDialog( FormClass : TFormClass) : Boolean
3281: //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3282: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3283: Procedure RestoreFormPlacement(Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3284: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3285: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3286: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3287: Function GetUniquefileNameInDir( const Path, FileNameMask : string) : string
3288: Function StrToInStr( const Str : string) : string
3289: Function InIstrToStr( const Str : string) : string
3290: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3291: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3292: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3293: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3294: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3295: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3296: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3297: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3298: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3299: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3300: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3301: Procedure AppTaskbarIcons( AppOnly : Boolean)
3302: Procedure InternalSaveGridLayout(Grid : TCustomGrid; IniFile : TObject; const Section : string)
3303: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3304: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3305: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3306: ***** uPSI_JvDBUtils;
3307: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3308: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3309: Procedure RefreshQuery( Query : TDataSet)
3310: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3311: Function DataSetSectionName( DataSet : TDataSet) : string
3312: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3313: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3314: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
Options: TLocateOptions) : Boolean

```

```

3315: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3316: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3317: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3318: Function ConfirmDelete : Boolean
3319: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3320: Procedure CheckRequiredField( Field : TField)
3321: Procedure CheckRequiredFields( const Fields : array of TField)
3322: Function DateToSQL( Value : TDateTime ) : string
3323: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string ) : string
3324: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string ) : string
3325: Function FormatSQLNumericRange( const FieldName:string;LowVal,HighVal,LowEmpty,
   HighEmpty:Double;Inclusive:Bool ):string
3326: Function StrMaskSQL( const Value : string ) : string
3327: Function FormatSQLCondition( const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool ):string
3328: Function FormatAnsiSQLCondition( const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool ):string
3329: Procedure _DBError( const Msg : string )
3330: Const ('TrueExpr','String '0=0
3331: Const ('sdfStandard16','String ''''''mm''/''dd''/''yyyy'''')
3332: Const ('sdfStandard32','String ''''''dd/mm/yyyy'''')
3333: Const ('sdfOracle','String ''TO_DATE('''dd/mm/yyyy'''', ''DD/MM/YYYY'')')
3334: Const ('sdfInterbase','String ''CAST('''mm''/''dd''/''yyyy''' AS DATE)')
3335: Const ('sdfMSSQL','String ''CONVERT(datetime, ''''mm''/''dd''/''yyyy'''', 103)')
3336: AddTypeS('Largeint', 'Longint'
3337: TIEException', '(ErNoImport, erCannotImport, erInvalidType, ErInternalError, '+
3338:   'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3339:   'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3340:   'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutOfRecordRange, '+
3341:   'erOutOfMemory,erException,erNullPointerException,erNullVariantError,erInterfaceNotSupportedError);
3342: (*-----*)
3343: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3344: begin
3345:   Function JIniReadBool( const FileName, Section, Line : string ) : Boolean
3346:   Function JIniReadInteger( const FileName, Section, Line : string ) : Integer
3347:   Function JIniReadString( const FileName, Section, Line : string ) : string
3348:   Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3349:   Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3350:   Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3351:   Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3352:   Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3353: end;
3354:
3355: (* === compile-time registration functions === *)
3356: (*-----*)
3357: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3358: begin
3359:   'UnixTimeStart','LongInt'( 25569 );
3360:   Function JEncodeDate( const Year : Integer; Month, Day : Word ) : TDateTime
3361:   Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word );
3362:   Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word );
3363:   Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer );
3364:   Function CenturyOfDate( const Date : TDateTime ) : Integer
3365:   Function CenturyBaseYear( const Date : TDateTime ) : Integer
3366:   Function DayOfDate( const Date : TDateTime ) : Integer
3367:   Function MonthOfDay( const Date : TDateTime ) : Integer
3368:   Function YearOfDate( const Date : TDateTime ) : Integer
3369:   Function JDayOfTheYear( const Date : TDateTime; var Year : Integer ) : Integer;
3370:   Function DayOfTheYear1( const Date : TDateTime ) : Integer;
3371:   Function DayOfTheYearToDate( const Year, Day : Integer ) : TDateTime
3372:   Function HourOfTime( const Date : TDateTime ) : Integer
3373:   Function MinuteOfTime( const Date : TDateTime ) : Integer
3374:   Function SecondOfTime( const Date : TDateTime ) : Integer
3375:   Function GetISOYearNumberOfDays( const Year : Word ) : Word
3376:   Function IsISOLongYear( const Year : Word ) : Boolean;
3377:   Function IsISOLongYear1( const Date : TDateTime ) : Boolean;
3378:   Function ISODayOfWeek( const Date : TDateTime ) : Word
3379:   Function JISOWeekNumber( Date: TDateTime; var YearOfWeekNumber, WeekDay: Integer ) : Integer;
3380:   Function ISOWeekNumber1( Date : TDateTime; var YearOfWeekNumber : Integer ) : Integer;
3381:   Function ISOWeekNumber2( Date : TDateTime ) : Integer;
3382:   Function ISOWeekToDate( const Year, Week, Day : Integer ) : TDateTime
3383:   Function JIsLeapYear( const Year : Integer ) : Boolean;
3384:   Function IsLeapYear1( const Date : TDateTime ) : Boolean;
3385:   Function JDaysInMonth( const Date : TDateTime ) : Integer
3386:   Function Make4DigitYear( Year, Pivot : Integer ) : Integer
3387:   Function JMakeYear4Digit( Year, WindowsillyYear : Integer ) : Integer
3388:   Function JEasterSunday( const Year : Integer ) : TDateTime // TDosDateTime', 'Integer
3389:   Function JFormatDateTime( Form : string; Date : TDateTime ) : string
3390:   Function FATDatesEqual( const FileTime1, FileTime2 : Int64 ) : Boolean;
3391:   Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime ) : Boolean;
3392:   Function HoursToMSEcs( Hours : Integer ) : Integer
3393:   Function MinutesToMSEcs( Minutes : Integer ) : Integer
3394:   Function SecondsToMSEcs( Seconds : Integer ) : Integer
3395:   Function TimeOfDateToSeconds( Date : TDateTime ) : Integer
3396:   Function TimeOfDateTimeToMSEcs( Date : TDateTime ) : Integer
3397:   Function DateTimeToLocalDateTime( Date : TDateTime ) : TDateTime
3398:   Function LocalDateTimeToDate( Date : TDateTime ) : TDateTime
3399:   Function DateTimeToDosDateTime( const Date : TDateTime ) : TDosDateTime
3400:   Function JDatetimeToFileTime( Date : TDateTime ) : TFileTime
3401:   Function JDatetimeToSystemTime( Date : TDateTime ) : TSystemTime;
3402:   Procedure DateTimeToSystemTime1( Date : TDateTime; var SysTime : TSystemTime );

```

```

3403: Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3404: Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3405: Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3406: Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3407: Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3408: Function DosDateTimeToStr( DateTime : Integer) : string
3409: Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3410: Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3411: Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3412: Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3413: Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3414: Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3415: Function FileTimeToStr( const FileTime : TFileTime) : string
3416: Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3417: Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3418: Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3419: Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3420: Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3421: Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3422: Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3423: TJclUnixTime32', 'Longword
3424: Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3425: Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3426: Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3427: Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3428: Function JNullStamp : TTimeStamp
3429: Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3430: Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3431: Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3432: Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3433: Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3434: Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3435: Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3436: Function LastWeekDay1( const Year, Month : Integer) : Integer;
3437: Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3438: Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3439: Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3440: Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3441: Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3442: Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3443: Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3444: Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3445: Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3446: FindClass('TOBJECT'), 'EJclDateTimeError
3447: end;
3448:
3449: procedure SIRegister_JclMiscel2(CL: TPSPPascalCompiler);
3450: begin
3451:   Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3452:   Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile:string): Boolean
3453:   Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3454:   Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3455:   Function WinExec32AndRedirectOutput(const Cmd : string; var Output: string; RawOutput:Boolean):Cardinal
3456:   TJclKillLevel', '( k1Normal, k1NoSignal, k1TimeOut )
3457:   Function ExitWindows( ExitCode : Cardinal) : Boolean
3458:   Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3459:   Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3460:   Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3461:   Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3462:   Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3463:   Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3464:   Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3465:   Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3466:   Function AbortShutDown : Boolean;
3467:   Function AbortShutDown1( const MachineName : string) : Boolean;
3468:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3469:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3470:   Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3471:   FindClass('TOBJECT'), 'EJclCreateProcessError
3472:   Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3473:   Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
Environment:PChar);
3474:   // with Add(EJclCreateProcessError) do
3475:   end;
3476:
3477:
3478: procedure SIRegister_JclAnsiStrings(CL: TPSPPascalCompiler);
3479: begin
3480:   //'\AnsiSigns','Set').SetSet(['-', '+']);
3481:   'C1_UPPER','LongWord( $0001';
3482:   'C1_LOWER','LongWord( $0002';
3483:   'C1_DIGIT','LongWord').SetUInt( $0004);
3484:   'C1_SPACE','LongWord').SetUInt( $0008);
3485:   'C1_PUNCT','LongWord').SetUInt( $0010);
3486:   'C1_CNTRL','LongWord').SetUInt( $0020);
3487:   'C1_BLANK','LongWord').SetUInt( $0040);
3488:   'C1_XDIGIT','LongWord').SetUInt( $0080);
3489:   'C1_ALPHA','LongWord').SetUInt( $0100);
3490:   AnsiChar', 'Char

```

```

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

```

```

3580: Function StrRestOf( const S : AnsiString; N : Integer ) : AnsiString
3581: Function StrRight( const S : AnsiString; Count : Integer ) : AnsiString
3582: Function CharEqualNoCase( const C1, C2 : AnsiChar ) : Boolean
3583: Function CharIsAlpha( const C : AnsiChar ) : Boolean
3584: Function CharIsAlphaNum( const C : AnsiChar ) : Boolean
3585: Function CharIsBlank( const C : AnsiChar ) : Boolean
3586: Function CharIsControl( const C : AnsiChar ) : Boolean
3587: Function CharIsDelete( const C : AnsiChar ) : Boolean
3588: Function CharIsDigit( const C : AnsiChar ) : Boolean
3589: Function CharIsLower( const C : AnsiChar ) : Boolean
3590: Function CharIsNumberChar( const C : AnsiChar ) : Boolean
3591: Function CharIsPrintable( const C : AnsiChar ) : Boolean
3592: Function CharIsPunctuation( const C : AnsiChar ) : Boolean
3593: Function CharIsReturn( const C : AnsiChar ) : Boolean
3594: Function CharIsSpace( const C : AnsiChar ) : Boolean
3595: Function CharIsUpper( const C : AnsiChar ) : Boolean
3596: Function CharIsWhiteSpace( const C : AnsiChar ) : Boolean
3597: Function CharType( const C : AnsiChar ) : Word
3598: Function CharHex( const C : AnsiChar ) : Byte
3599: Function CharLower( const C : AnsiChar ) : AnsiChar
3600: Function CharUpper( const C : AnsiChar ) : AnsiChar
3601: Function CharToggleCase( const C : AnsiChar ) : AnsiChar
3602: Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3603: Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3604: Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer ) : Integer
3605: Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar ) : Integer
3606: Procedure StrToStrings(S,Sep:AnsiString; const List:TStrings;const AllowEmptyString : Boolean)
3607: Procedure StrToStrings(S,Sep: AnsiString; const List:TStrings;const AllowEmptyString : Boolean)
3608: Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3609: Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean )
3610: Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean )
3611: Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean )
3612: Function AddStringToStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3613: Function BooleanToStr( B : Boolean ) : AnsiString
3614: Function FileToString( const FileName : AnsiString ) : AnsiString
3615: Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean )
3616: Procedure StringToFile2( const S, FileName : string );
3617: Function StrToken( var S : AnsiString; Separator : AnsiChar ) : AnsiString
3618: Procedure StrTokens( const S : AnsiString; const List : TStrings )
3619: Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings )
3620: //Function StrWord( var S : PAnsiChar; out Word : AnsiString ) : Boolean
3621: Function StrToFloatSafe( const S : AnsiString ) : Float
3622: Function StrToIntSafe( const S : AnsiString ) : Integer
3623: Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer );
3624: Function ArrayOf( List : TStrings ) : TDynStringArray;
3625: EJclStringError', 'EJclError
3626: function IsClass(Address: TObject): Boolean;
3627: function IsObject(Address: TObject): Boolean;
3628: // Console Utilities
3629: //function ReadKey: Char;
3630: function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3631: function JclGUIDToString(const GUID: TGUID): string;
3632: function JclStringToGUID(const S: string): TGUID;
3633: end;
3634:
3635:
3636: ****uPSI_JvDBUtil;
3637: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3638: Function GetQueryResult( const DatabaseName, SQL : string ) : Variant
3639: Function GetStoredProcedureResult( const ADatabaseName, AStoredProcedureName : string; AParams : array of Variant; const AResultName : string ) : Variant
3640: //Function StrFieldDesc( Field : FLDDesc ) : string
3641: Function Var2Type( V : Variant; const VarType : Integer ) : Variant
3642: Procedure CopyRecord( DataSet : TDataSet )
3643: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string; MasterField : Word; ModOp, DelOp : RINTQual )
3644: Procedure AddMasterPassword( Table : TTable; pswd : string )
3645: Procedure PackTable( Table : TTable )
3646: Procedure PackEncryptedTable( Table : TTable; pswd : string )
3647: Function EncodeQuotes( const S : string ) : string
3648: Function Cmp( const S1, S2 : string ) : Boolean
3649: Function SubStr( const S : string; const Index : Integer; const Separator : string ) : string
3650: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string ) : string
3651: Function ReplaceString( S : string; const OldPattern, NewPattern : string ) : string
3652: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer )
3653: *****uPSI_JvJvBDEUtils;*****
3654: //JvBDEUtils
3655: Function CreateDbLocate : TJvLocateObject
3656: //Function CheckOpen( Status : DBIResult ) : Boolean
3657: Procedure FetchAllRecords( DataSet : TBDEDataSet )
3658: Function TransActive( Database : TDatabase ) : Boolean
3659: Function AsyncQrySupported( Database : TDatabase ) : Boolean
3660: Function GetQuoteChar( Database : TDatabase ) : string
3661: Procedure ExecuteQuery( const DbName, QueryText : string )
3662: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string )
3663: Function FieldLogicMap( FldType : TFieldType ) : Integer
3664: Function FieldSubtypeMap( FldType : TFieldType ) : Integer Value : string; Buffer : Pointer)
3665: Function GetAliasPath( const AliasName : string ) : string

```

```

3666: Function IsDirectory( const DatabaseName : string) : Boolean
3667: Function GetBdeDirectory : string
3668: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3669: Function DataSetFindValue( ADataSet : TBDEDataset; const Value, FieldName : string) : Boolean
3670: Function DataSetFindLike( ADataSet : TBDEDataset; const Value, FieldName : string) : Boolean
3671: Function DataSetRecNo( DataSet : TDataSet) : Longint
3672: Function DataSetRecordCount( DataSet : TDataSet) : Longint
3673: Function DataSetPositionStr( DataSet : TDataSet) : string
3674: Procedure DataSetShowDeleted( DataSet : TBDEDataset; Show : Boolean)
3675: Function CurrentRecordDeleted( DataSet : TBDEDataset) : Boolean
3676: Function IsFilterApplicable( DataSet : TDataSet) : Boolean
3677: Function IsBookmarkStable( DataSet : TBDEDataset) : Boolean
3678: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3679: Procedure RestoreIndex( Table : TTable)
3680: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3681: Procedure PackTable( Table : TTable)
3682: Procedure ReindexTable( Table : TTable)
3683: Procedure BdeFlushBuffers
3684: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3685: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3686: Procedure DbNotSupported
3687: Procedure ExportDataSet( Source : TBDEDataset; DestTable : TTable; TableType : TTableType; const
  AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3688: Procedure ExportDataSetEx( Source : TBDEDataset; DestTable : TTable; TableType : TTableType; const
  AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3689: Procedure
  ImportDataSet(Source:TBDEDataset;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3690: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3691: ****uPSI_JvDateUtil;
3692: function CurrentYear: Word;
3693: function IsLeapYear(AYear: Integer): Boolean;
3694: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3695: function FirstDayOfPrevMonth: TDateTime;
3696: function LastDayOfPrevMonth: TDateTime;
3697: function FirstDayOfNextMonth: TDateTime;
3698: function ExtractDay(ADate: TDateTime): Word;
3699: function ExtractMonth(ADate: TDateTime): Word;
3700: function ExtractYear(ADate: TDateTime): Word;
3701: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3702: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3703: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3704: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3705: function ValidateDate(ADate: TDateTime): Boolean;
3706: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3707: function MonthsBetween(Date1, Date2: TDateTime): Double;
3708: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3709: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3710: function DaysBetween(Date1, Date2: TDateTime): Longint;
3711: { The same as previous but if Date2 < Date1 result = 0 }
3712: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3713: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3714: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3715: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3716: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3717: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3718: { String to date conversions }
3719: function GetDateOrder(const DateFormat: string): TDateOrder;
3720: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3721: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3722: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3723: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3724: function DefDateFormat(FourDigitYear: Boolean): string;
3725: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3726: -----
3727: ***** JvUtils;*****
3728: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3729: function GetWordOnPos(const S: string; const P: Integer): string;
3730: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3731: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3732: { SubStr returns substring from string, S, separated with Separator string}
3733: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3734: { SubStrEnd same to previous function but Index numerated from the end of string }
3735: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3736: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3737: function SubWord(P: PChar; var P2: PChar): string;
3738: { NumberByWord returns the text representation of
  the number, N, in normal russian language. Was typed from Monitor magazine }
3739: function NumberByWord(const N: Longint): string;
3740: // function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3741: //the symbol Pos is pointed. Lines separated with #13 symbol }
3742: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3743: { GetXYByPos is same to previous function, but returns X position in line too}
3744: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3745: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3746: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3747: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3748: function ConcatSep(const S, S2, Separator: string): string;
3749: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3750: function ConcatLeftSep(const S, S2, Separator: string): string;

```

```

3752: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3753: function MinimizeString(const S: string; const MaxLen: Integer): string;
3754: { Next 4 function for russian chars transliterating.
3755:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3756: procedure Dos2Win(var S: string);
3757: procedure Win2Dos(var S: string);
3758: function Dos2WinRes(const S: string): string;
3759: function Win2DosRes(const S: string): string;
3760: function Win2Koi(const S: string): string;
3761: { Spaces returns string consists on N space chars }
3762: function Spaces(const N: Integer): string;
3763: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3764: function AddSpaces(const S: string; const N: Integer): string;
3765: { function LastDate for russian users only } { returns date relative to current date: '' }
3766: function LastDate(const Dat: TDateTime): string;
3767: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3768: function CurrencyToStr(const Cur: currency): string;
3769: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3770: function Cmp(const S1, S2: string): Boolean;
3771: { StringCat add S2 string to S1 and returns this string }
3772: function StringCat(var S1: string; S2: string): string;
3773: { HasChar returns True, if Char, Ch, contains in string, S }
3774: function HasChar(const Ch: Char; const S: string): Boolean;
3775: function HasAnyChar(const Chars: string; const S: string): Boolean;
3776: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3777: function CountOfChar(const Ch: Char; const S: string): Integer;
3778: function DefStr(const S: string; Default: string): string;
3779: {**** files routines}
3780: { GetWinDir returns Windows folder name }
3781: function GetWinDir: TFileName;
3782: function GetSysDir: String;
3783: { GetTempDir returns Windows temporary folder name }
3784: function GetTempDir: string;
3785: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3786: function GenTempFileName(FileName: string): string;
3787: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3788: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3789: { ClearDir clears folder Dir }
3790: function ClearDir(const Dir: string): Boolean;
3791: { DeleteDir clears and then delete folder Dir }
3792: function DeleteDir(const Dir: string): Boolean;
3793: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3794: function FileEquMask(FileName, Mask: TFileName): Boolean;
3795: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3796:   Masks must be separated with comma ('') }
3797: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3798: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3799: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3800: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3801: { FileGetInfo fills SearchRec record for specified file attributes}
3802: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3803: { HasSubFolder returns True, if folder APath contains other folders }
3804: function HasSubFolder(APath: TFileName): Boolean;
3805: { IsEmptyFolder returns True, if there are no files or folders in given folder, APPath}
3806: function IsEmptyFolder(APath: TFileName): Boolean;
3807: { AddSlash add slash Char to Dir parameter, if needed }
3808: procedure AddSlash(var Dir: TFileName);
3809: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3810: function AddSlash2(const Dir: TFileName): string;
3811: { AddPath returns FileName with Path, if FileName not contain any path }
3812: function AddPath(const FileName, Path: TFileName): TFileName;
3813: function AddPaths(const PathList, Path: string): string;
3814: function ParentPath(const Path: TFileName): TFileName;
3815: function FindInPath(const FileName, PathList: string): TFileName;
3816: function FindInPaths(const fileName, paths: String): String;
3817: {$_IFDEF BCB1}
3818: { BrowseForFolder displays Browse For Folder dialog }
3819: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3820: {$_ENDIF BCB1}
3821: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
AHelpContext : THelpContext) : Boolean
3822: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
AHelpContext : THelpContext) : Boolean
3823: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3824: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3825:
3826: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3827: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3828: { HasParam returns True, if program running with specified parameter, Param }
3829: function HasParam(const Param: string): Boolean;
3830: function HasSwitch(const Param: string): Boolean;
3831: function Switch(const Param: string): string;
3832: { ExePath returns ExtractFilePath(ParamStr(0)) }
3833: function ExePath: TFileName;
3834: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3835: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3836: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3837: {**** Graphic routines }
3838: { TTFontSelected returns True, if True Type font is selected in specified device context }

```

```

3839: function TTFontSelected(const DC: HDC): Boolean;
3840: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3841: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3842: {**** Windows routines }
3843: { SetWindowTop put window to top without recreating window }
3844: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3845: {**** other routines }
3846: { KeyPressed returns True, if Key VK is now pressed }
3847: function KeyPressed(VK: Integer): Boolean;
3848: procedure SwapInt(var Int1, Int2: Integer);
3849: function IntPower(Base, Exponent: Integer): Integer;
3850: function ChangeTopException(E: TObject): TObject;
3851: function StrToBool(const S: string): Boolean;
3852: {$IFDEF COMPILER3_UP}
3853: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3854:   Length of MaxLen bytes. The compare operation is controlled by the
3855:   current Windows locale. The return value is the same as for CompareStr. }
3856: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3857: function AnsiStrICmp(S1, S2: PChar): Integer;
3858: {$ENDIF}
3859: function Var2Type(V: Variant; const VarType: Integer): Variant;
3860: function VarToInt(V: Variant): Integer;
3861: function VarToFloat(V: Variant): Double;
3862: { following functions are not documented because they are don't work properly , so don't use them }
3863: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3864: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3865: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3866: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3867: function GetParameter: string;
3868: function GetLongFileName(FileName: string): string;
3869: {* from FileCtrl}
3870: function DirectoryExists(const Name: string): Boolean;
3871: procedure ForceDirectories(Dir: string);
3872: {# from FileCtrl}
3873: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3874: function GetComputerID: string;
3875: function GetComputerName: string;
3876: {**** string routines }
3877: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
3878: same Index.Also see RAUtils.ReplaceSokr function }
3879: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3880: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3881:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with the
3882:   same Index, and then update NewSelStart variable }
3883: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3884: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3885: function CountOfLines(const S: string): Integer;
3886: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3887: procedure DeleteEmptyLines(Ss: TStrings);
3888: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3889:   Note: If strings SQL alriady contains where-statement, it must be started on begining of any line }
3890: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3891: {**** files routines - }
3892: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3893:   Resource can be compressed using MS Compress program}
3894: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3895: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool;
3896: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3897: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3898: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3899: { IniReadSection read section, Section, from ini-file,
3900:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3901:   Note: TiniFile.ReadSection function reads only strings with '=' symbol.}
3902: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3903: { LoadTextFile load text file, FileName, into string }
3904: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3905: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList}
3906: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3907: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3908: {$IFDEF COMPILER3_UP}
3909: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3910: function TargetFileName(const FileName: TFileName): TFileName;
3911: { return filename ShortCut linked to }
3912: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3913: {$ENDIF COMPILER3_UP}
3914: { LoadIconToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3915: procedure LoadIconToImage(ALarge, ASmall: TImageList; const NameRes: string);
3916: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3917: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3918: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3919: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3920: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3921: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3922: { Cinema draws some visual effect }
3923: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}): TRect;
3924: { Roughed fills rect with special 3D pattern }
3925: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);

```

```

3926: { 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 }
3927: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3928: { TextWidth calculate text width for writing using standard desktop font }
3929: function TextWidth(AString: string): Integer;
3930: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3931: function DefineCursor(Identifier: PChar): TCursor;
3932: {**** other routines - }
3933: { FindFormByClass returns first form specified class, FormClass, owned by Application global variable }
3934: function FindFormByClass(FormClass: TFormClass): TForm;
3935: function FindFormByName(FormClassName: string): TForm;
3936: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
      having Tag property value, equaled to Tag parameter }
3937: function FindByTag(WinControl: TWinControl; ComponentClass: TComponentClass; const Tag: Integer): TComponent;
3938: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3940: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3941: { RBTAG searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3942: function RBTAG(Parent: TWinControl): Integer;
3943: { AppMinimized returns True, if Application is minimized }
3944: function AppMinimized: Boolean;
3945: { MessageBox is Application.MessageBox with string (not PChar) parameters.
      if Caption parameter = '', it replaced with Application.Title }
3946: function MessageBoxJ(const Msg: string; Caption: string; const Flags: Integer): Integer;
3947: function MsgBox2(const Msg, ACaption: string; DlgType: TMsgDlgType);
3948: function MsgBox2(const Msg, ACaption: string; DlgType: TMsgDlgType);
3949: Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3950: function MsgBoxDef(const Msg, ACaption: string; DlgType: TMsgDlgType);
3951: Buttons: TMsgDlgButtons; DefButton: TMsgDlgBtn; HelpContext: Integer; Control: TWinControl): Integer;
3952: { Delay stop program execution to MSec msec }
3953: procedure Delay(MSec: Longword);
3954: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3955: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3956: procedure EnableMenuItem(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3957: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3958: function PanelBorder(Panel: TCustomPanel): Integer;
3959: function Pixels(Control: TControl; APixels: Integer): Integer;
3960: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3961: procedure Error(const Msg: string);
3962: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
      const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3963: {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3964: const HideSelColor: Boolean;
3965: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
      const HideSelColor: Boolean): string;
3966: function ItemHtWidth(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
      const HideSelColor: Boolean): Integer;
3967: function ItemHtPlain(const Text: string): string;
3968: { ClearList - clears list of TObject }
3969: procedure ClearList(List: TList);
3970: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
3971: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3972: { RTTI support }
3973: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3974: function GetPropStr(Obj: TObject; const PropName: string): string;
3975: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3976: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3977: procedure PrepareIniSection(SS: TStrings);
3978: { following functions are not documented because they are don't work properly, so don't use them }
3979: {$IFDEF COMPILER2}
3980: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3981: {$ENDIF}
3982: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3983: begin
3984:   Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3985:   Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3986:   Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3987:   Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3988:   Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3989:   Procedure BoxSetItem( List : TWinControl; Index : Integer)
3990:   Function BoxGetFirstSelection( List : TWinControl ) : Integer
3991:   Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3992: end;
3993: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3994: begin
3995:   Const ('MaxInitStrNum','LongInt' ( 9));
3996:   Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar: AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3997:   Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3998:   Function JvAnsiStrSplitStrings(const InStr: AnsiString;const SplitChar,
      QuoteChar: AnsiChar;OutStrs:TStrings):Integer;
3999:   Function JvAnsiStrStrip( S : AnsiString ) : AnsiString
4000:   Function JvStrStrip( S : string ) : string
4001:   Function GetString( var Source : AnsiString; const Separator : AnsiString ) : AnsiString
4002:   Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar ) : AnsiString
4003:   Function BuildPathName( const PathName, FileName : AnsiString ) : AnsiString
4004:   Function StrEatWhiteSpace( const S : string ) : string
4005:   Function HexToAscii( const S : AnsiString ) : AnsiString

```

```

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

```

```

4099: { NPos searches for a N-th position of substring C in a given string. }
4100: function MakeStr(C: Char; N: Integer): string;
4101: function MS(C: Char; N: Integer): string;
4102: { MakeStr return a string of length N filled with character C. }
4103: function AddChar(C: Char; const S: string; N: Integer): string;
4104: { AddChar return a string left-padded to length N with characters C. }
4105: function AddCharR(C: Char; const S: string; N: Integer): string;
4106: { AddCharR return a string right-padded to length N with characters C. }
4107: function LeftStr(const S: string; N: Integer): string;
4108: { LeftStr return a string right-padded to length N with blanks. }
4109: function RightStr(const S: string; N: Integer): string;
4110: { RightStr return a string left-padded to length N with blanks. }
4111: function CenterStr(const S: string; Len: Integer): string;
4112: { CenterStr centers the characters in the string based upon the Len specified. }
4113: function CompStr(const S1, S2: string): Integer;
4114: { CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4115: function CompText(const S1, S2: string): Integer;
4116: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4117: function Copy2Symb(const S: string; Symb: Char): string;
4118: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4119: function Copy2SymbDel(var S: string; Symb: Char): string;
4120: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4121: function Copy2Space(const S: string): string;
4122: { Copy2Space returns a substring of a string S from beginning to first white space. }
4123: function Copy2SpaceDel(var S: string): string;
4124: { Copy2SpaceDel returns a substring of a string S from beginning to first
4125:   white space and removes this substring from S. }
4126: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4127: { Returns string, with the first letter of each word in uppercase,
4128:   all other letters in lowercase. Words are delimited by WordDelims. }
4129: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4130: { WordCount given a set of word delimiters, returns number of words in S. }
4131: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4132: { Given a set of word delimiters, returns start position of N'th word in S. }
4133: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4134: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TCharSet; var Pos: Integer): string;
4135: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4136: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4137:   delimiters, return the N'th word in S. }
4138: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4139: { ExtractSubstr given set of word delimiters, returns the substring from S,that started from position Pos.}
4140: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4141: { IsWordPresent given set of word delimiters, returns True if word W is present in string S. }
4142: function QuotedString(const S: string; Quote: Char): string;
4143: { QuotedString returns given string as a quoted string, using the provided Quote character. }
4144: function ExtractQuotedString(const S: string; Quote: Char): string;
4145: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4146:   and reduces pairs of Quote characters within the quoted string to a single character. }
4147: function FindPart(const HelpWilds, InputStr: string): Integer;
4148: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4149: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4150: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4151: function XorString(const Key, Src: ShortString): ShortString;
4152: function XorEncode(const Key, Source: string): string;
4153: function XorDecode(const Key, Source: string): string;
4154: { ** Command line routines ** }
4155: {$IFNDEF COMPILER4_UP}
4156: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet; IgnoreCase: Boolean): Boolean;
4157: {$ENDIF}
4158: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4159: { ** Numeric string handling routines ** }
4160: function Numb2USA(const S: string): string;
4161: { Numb2USA converts numeric string S to USA-format. }
4162: function Dec2Hex(N: Longint; A: Byte): string;
4163: function D2H(N: Longint; A: Byte): string;
4164: { Dec2Hex converts the given value to a hexadecimal string representation
4165:   with the minimum number of digits (A) specified. }
4166: function Hex2Dec(const S: string): Longint;
4167: function H2D(const S: string): Longint;
4168: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4169: function Dec2Numb(N: Longint; A, B: Byte): string;
4170: { Dec2Numb converts the given value to a string representation with the
4171:   base equal to B and with the minimum number of digits (A) specified. }
4172: function Numb2Dec(S: string; B: Byte): Longint;
4173: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4174: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4175: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4176: function IntToRoman(Value: Longint): string;
4177: { IntToRoman converts the given value to a roman numeric string representation. }
4178: function RomanToInt(const S: string): Longint;
4179: { RomanToInt converts the given string to an integer value. If the string
4180:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4181: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4182: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4183: ***** JvFileUtil*****
4184: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4185: procedure CopyFileEx(const FileName, DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:TControl);
4186: procedure MoveFile(const FileName, DestName: TFileName);

```

```

4187: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4188: {$IFDEF_COMPILER4_UP}
4189: function GetFileSize(const FileName: string): Int64;
4190: {$ELSE}
4191: function GetFileSize(const FileName: string): Longint;
4192: {$ENDIF}
4193: function FileDateTime(const FileName: string): TDateTime;
4194: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4195: function DeleteFiles(const FileMask: string): Boolean;
4196: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4197: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4198: function NormalDir(const DirName: string): string;
4199: function RemoveBackSlash(const DirName: string): string;
4200: function ValidfileName(const FileName: string): Boolean;
4201: function DirExists(Name: string): Boolean;
4202: procedure ForceDirectories(Dir: string);
4203: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4204: {$IFDEF_COMPILER4_UP} overload; {$ENDIF}
4205: {$IFDEF_COMPILER4_UP}
4206: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4207: {$ENDIF}
4208: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4209: {$IFDEF_COMPILER4_UP} overload; {$ENDIF}
4210: {$IFDEF_COMPILER4_UP}
4211: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4212: {$ENDIF}
4213: function GetTempDir: string;
4214: function GetWindowsDir: string;
4215: function GetSystemDir: string;
4216: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4217: {$IFDEF_WIN32}
4218: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4219: function ShortToLongFileName(const ShortName: string): string;
4220: function ShortToLongPath(const ShortName: string): string;
4221: function LongToShortFileName(const LongName: string): string;
4222: function LongToShortPath(const LongName: string): string;
4223: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4224: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4225: {$ENDIF_WIN32}
4226: {$IFNDEF_COMPILER3_UP}
4227: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4228: {$ENDIF}
4229: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext ) : TJvCalculatorForm;
4230: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode ) : TWinControl;
4231: Function CreatePopupCalculator( AOwner : TComponent ) : TWinControl;
4232: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean );
4233:
4234: //*****procedure SIRegister_VarHlpr(CL: TPSPPascalCompiler);
4235: Procedure VariantClear( var V : Variant);
4236: Procedure VariantArrayRedim( var V : Variant; High : Integer);
4237: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer);
4238: Procedure VariantCpy( const src : Variant; var dst : Variant);
4239: Procedure VariantAdd( const src : Variant; var dst : Variant);
4240: Procedure VariantSub( const src : Variant; var dst : Variant);
4241: Procedure VariantMul( const src : Variant; var dst : Variant);
4242: Procedure VariantDiv( const src : Variant; var dst : Variant);
4243: Procedure VariantMod( const src : Variant; var dst : Variant);
4244: Procedure VariantAnd( const src : Variant; var dst : Variant);
4245: Procedure VariantOr( const src : Variant; var dst : Variant);
4246: Procedure VariantXor( const src : Variant; var dst : Variant);
4247: Procedure VariantShl( const src : Variant; var dst : Variant);
4248: Procedure VariantShr( const src : Variant; var dst : Variant);
4249: Function VariantAdd2( const V1 : Variant; const V2 : Variant ) : Variant;
4250: Function VariantSub2( const V1 : Variant; const V2 : Variant ) : Variant;
4251: Function VariantMul2( const V1 : Variant; const V2 : Variant ) : Variant;
4252: Function VariantDiv2( const V1 : Variant; const V2 : Variant ) : Variant;
4253: Function VariantMod2( const V1 : Variant; const V2 : Variant ) : Variant;
4254: Function VariantAnd2( const V1 : Variant; const V2 : Variant ) : Variant;
4255: Function VariantOr2( const V1 : Variant; const V2 : Variant ) : Variant;
4256: Function VariantXor2( const V1 : Variant; const V2 : Variant ) : Variant;
4257: Function VariantShl2( const V1 : Variant; const V2 : Variant ) : Variant;
4258: Function VariantShr2( const V1 : Variant; const V2 : Variant ) : Variant;
4259: Function VariantNot( const V1 : Variant ) : Variant;
4260: Function VariantNeg( const V1 : Variant ) : Variant;
4261: Function VariantGetElement( const V : Variant; i1 : integer ) : Variant;
4262: Function VariantGetElement1( const V : Variant; i1, i2 : integer ) : Variant;
4263: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer ) : Variant;
4264: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer ) : Variant;
4265: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer ) : Variant;
4266: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer );
4267: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer );
4268: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer );
4269: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer );
4270: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer );
4271: end;
4272:
4273: *****unit uPSI_JvgUtils;*****
4274: function IsEven(I: Integer): Boolean;
4275: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;

```

```

4276: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4277: procedure SwapInt(var I1, I2: Integer);
4278: function Spaces(Count: Integer): string;
4279: function DupStr(const Str: string; Count: Integer): string;
4280: function DupChar(C: Char; Count: Integer): string;
4281: procedure Msg(const AMsg: string);
4282: function RectW(R: TRect): Integer;
4283: function RectH(R: TRect): Integer;
4284: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4285: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4286: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4287: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4288:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4289: procedure DrawTextInRect(DC: HDC; R: TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4290: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4291:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor, HighlightColor, ShadowColor: TColor;
4292:   Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4293: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4294: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4295:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean;BackgrColor:Longint;ATransparent: Boolean): TRect;
4296: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4297: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4298: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4299:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y_in_rect!
4300:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4301: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4302:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y_in_rect!
4303:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4304:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4305: procedure BringParentWindowToFront(Wnd: TWinControl);
4306: function GetParentForm(Control: TControl): TForm;
4307: procedure GetWindowImageFrom(Control: TWinControl; X,Y:Integer;ADrawSelf,ADrawChildWindows:Bool;DC:HDC);
4308: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4309: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4310: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4311: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4312: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4313: function CalcMathString(AExpression: string): Single;
4314: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4315: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4316: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4317: procedure TypeStringOnKeyboard(const S: string);
4318: function NextStringGridCell(Grid: TStringGrid): Boolean;
4319: procedure DrawTextExtAligned(Canvas:TCanvas;const Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Bool);
4320: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4321: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4322: function ComponentToString(Component: TComponent): string;
4323: procedure StringToComponent(Component: TComponent; const Value: string);
4324: function PlayWaveResource(const ResName: string): Boolean;
4325: function UserName: string;
4326: function ComputerName: string;
4327: function CreateIniFileName: string;
4328: function ExpandString(const Str: string; Len: Integer): string;
4329: function Transliterate(const Str: string; RusToLat: Boolean): string;
4330: function IsSmallFonts: Boolean;
4331: function SystemColorDepth: Integer;
4332: function GetFileTypeJ(const FileName: string): TglFileType;
4333: Function GetFileType( hFile : THandle ) : DWORD';
4334: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4335: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4336:
4337: { **** Utility routines of unit classes }
4338: function LineStart(Buffer, BufPos: PChar): PChar;
4339: function ExtractStrings(Separators,WhiteSpace: TSysCharSet; Content: PChar; '+
4340:   'Strings: TStrings): Integer
4341: Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4342: Procedure RegisterClass( AClass : TPersistentClass)
4343: Procedure RegisterClasses( AClasses : array of TPersistentClass)
4344: Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4345: Procedure UnRegisterClass( AClass : TPersistentClass)
4346: Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4347: Procedure UnRegisterModuleClasses( Module : HMODULE)
4348: Function FindGlobalComponent( const Name : string) : TComponent
4349: Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4350: Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4351: Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4352: Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4353: Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4354: Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4355: Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4356: Procedure GlobalFixupReferences
4357: Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4358: Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4359: Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4360: Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4361: Procedure RemoveFixups( Instance : TPersistent)
4362: Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4363: Procedure BeginGlobalLoading

```

```

4364: Procedure NotifyGlobalLoading
4365: Procedure EndGlobalLoading
4366: Function GetUltimateOwner1( ACollection : TCollection ) : TPersistent;
4367: Function GetUltimateOwner( APersistent : TPersistent ) : TPersistent;
4368: // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4369: //Function MakeObjectInstance( Method : TWndMethod ) : Pointer
4370: Procedure FreeObjectInstance( ObjectInstance : Pointer)
4371: // Function AllocateHWnd( Method : TWndMethod ) : HWnd
4372: Procedure DeAllocateHWnd( Wnd : HWnd )
4373: Function AncestorisValid( Ancestor : TPersistent; Root, RootAncestor : TComponent ) : Boolean
4374: *****unit uPSI_SqlTimSt and DB;*****
4375: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLOTimeStamp );
4376: Function VarSQLTimeStampCreate3: Variant;
4377: Function VarSQLTimeStampCreate2( const AValue : string ) : Variant;
4378: Function VarSQLTimeStampCreate1( const AValue : TDateTime ) : Variant;
4379: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLOTimeStamp ) : Variant;
4380: Function VarSQLTimeStamp : TVarType
4381: Function VarIsSQLTimeStamp( const aValue : Variant ) : Boolean;
4382: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLOTimeStamp ) : TSQLOTimeStamp //beta
4383: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLOTimeStamp ) : TSQLOTimeStamp //beta
4384: Function VarToSQLTimeStamp( const aValue : Variant ) : TSQLOTimeStamp
4385: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLOTimeStamp ) : string
4386: Function SQLDayOfWeek( const DateTime : TSQLOTimeStamp ) : integer
4387: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime ) : TSQLOTimeStamp
4388: Function SQLTimeStampToDateTime( const DateTime : TSQLOTimeStamp ) : TDateTime
4389: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLOTimeStamp ) : Boolean
4390: Function StrToSQLTimeStamp( const S : string ) : TSQLOTimeStamp
4391: Procedure CheckSQLTimeStamp( const ASQLTimeStamp : TSQLOTimeStamp )
4392: Function ExtractFieldName( const Fields : string; var Pos : Integer ) : string;
4393: Function ExtractFieldName( const Fields : WideString; var Pos : Integer ) : WideString;
4394: //Procedure RegisterFields( const FieldClasses : array of TFieldClass )
4395: Procedure DatabaseError( const Message : WideString; Component : TComponent )
4396: Procedure DatabaseErrorFmt( const Message:WideString; const Args:array of const;Component:TComponent )
4397: Procedure DisposeMem( var Buffer, Size : Integer )
4398: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer ) : Boolean
4399: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4400: Function VarTypeToDataType( VarType : Integer ) : TFieldType
4401: *****unit JvStrings;*****
4402: {template functions}
4403: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4404: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4405: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4406: function RemoveMasterBlocks(const SourceStr: string): string;
4407: function RemoveFields(const SourceStr: string): string;
4408: {http functions}
4409: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4410: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4411: {set functions}
4412: procedure SplitSet(AText: string; AList: TStringList);
4413: function JoinSet(AList: TStringList): string;
4414: function FirstOfSet(const AText: string): string;
4415: function LastOfSet(const AText: string): string;
4416: function CountOfSet(const AText: string): Integer;
4417: function SetRotateRight(const AText: string): string;
4418: function SetRotateLeft(const AText: string): string;
4419: function SetPick(const AText: string; AIIndex: Integer): string;
4420: function SetSort(const AText: string): string;
4421: function SetUnion(const Set1, Set2: string): string;
4422: function SetIntersect(const Set1, Set2: string): string;
4423: function SetExclude(const Set1, Set2: string): string;
4424: {replace any <,> etc by &lt;,&gt;}
4425: function XMLSafe(const AText: string): string;
4426: {simple hash, Result can be used in Encrypt}
4427: function Hash(const AText: string): Integer;
4428: { Base64 encode and decode a string }
4429: function B64Encode(const S: AnsiString): AnsiString;
4430: function B64Decode(const S: AnsiString): AnsiString;
4431: {Basic encryption from a Borland Example}
4432: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4433: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4434: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4435: function EncryptB64(const InString:AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4436: function DecryptB64(const InString:AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4437: procedure CSVToTags(Src, Dst: TStringList);
4438: // converts a csv list to a tagged string list
4439: procedure TagsToCSV(Src, Dst: TStringList);
4440: // converts a tagged string list to a csv list
4441: // only fieldnames from the first record are scanned in the other records
4442: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4443: {selects akey=value from Src and returns recordset in Dst}
4444: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4445: {filters Src for akey=value}
4446: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4447: {orders a tagged Src list by akey}
4448: function PosStr(const FindString, SourceString: string;
4449: StartPos: Integer = 1): Integer;
4450: { PosStr searches the first occurrence of a substring FindString in a string
4451: given by SourceString with case sensitivity (upper and lower case characters
4452: are differed). This function returns the index value of the first character

```

```

4453:   of a specified substring from which it occurs in a given string starting with
4454:   StartPos character index. If a specified substring is not found _PosStr
4455:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings from www.torry.ru). }
4456: function PosStrLast(const FindString, SourceString: string): Integer;
4457: {finds the last occurrence}
4458: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4459: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4460: { PosText searches the first occurrence of a substring FindString in a string
4461:   given by SourceString without case sensitivity (upper and lower case characters are not differed). This
4462:   function returns the index value of the first character of a specified substring from which it occurs in a
4463:   given string starting with Start
4464: function PosTextLast(const FindString, SourceString: string): Integer;
4465: {finds the last occurrence}
4466: function NameValuesToXML(const AText: string): string;
4467: {$IFDEF MSWINDOWS}
4468: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4469: {$ENDIF MSWINDOWS}
4470: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4471: procedure SaveString(const AFile, AText: string);
4472: Procedure SaveStringasFile( const AFile, AText : string)
4473: function LoadStringJ(const AFile: string): string;
4474: Function LoadStringofFile( const AFile : string) : string
4475: Procedure SaveStringToFile( const AFile, AText : string)
4476: Function LoadStringFromFile( const AFile : string) : string
4477: function HexToColor(const AText: string): TColor;
4478: function UppercaseHTMLTags(const AText: string): string;
4479: function LowercaseHTMLTags(const AText: string): string;
4480: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4481: function RelativePath(const ASrc, ADst: string): string;
4482: function GetToken(var Start: Integer; const SourceText: string): string;
4483: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4484: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4485: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4486: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4487: // parses the beginning of an attribute: space + alpha character
4488: function ParseAttribute(var Start: Integer; const SourceText: string; var AName, AValue: string): Boolean;
4489: // parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4490: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4491: // parses all name=value attributes to the attributes TStringList
4492: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4493: // checks if a name="value" pair exists and returns any value
4494: function GetStrValue(const AText, AName, ADefault: string): string;
4495: // retrieves string value from a line like:
4496: // name="jan verhoeven" email="janl dott verhoeven att wxs dott nl"
4497: // returns ADefault when not found
4498: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4499: // same for a color
4500: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4501: // same for an Integer
4502: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4503: // same for a float
4504: function GetBoolValue(const AText, AName: string): Boolean;
4505: // same for Boolean but without default
4506: function GetValue(const AText, AName: string): string;
4507: // retrieves string value from a line like: name="jan verhoeven" email="janl verhoeven att wxs dott nl"
4508: procedure SetValue(var AText: string; const AName, AValue: string);
4509: // sets a string value in a line
4510: procedure DeleteValue(var AText: string; const AName: string);
4511: // deletes a AName="value" pair from AText
4512: procedure GetNames(AText: string; AList: TStringList);
4513: // get a list of names from a string with name="value" pairs
4514: function GetHTMLColor(AColor: TColor): string;
4515: // converts a color value to the HTML hex value
4516: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4517: // finds a string backward case sensitive
4518: function BackPostText(Start: Integer; const FindString, SourceString: string): Integer;
4519: // finds a string backward case insensitive
4520: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4521: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4522: // finds a text range, e.g. <TD>....</TD> case sensitive
4523: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);
4524: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4525: // finds a text range, e.g. <TD>....</td> case insensitive
4526: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4527: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4528: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4529: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);
4530: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4531: // finds a text range backward, e.g. <TD>....</td> case insensitive
4532: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4533: var RangeEnd: Integer): Boolean;
4534: // finds a HTML or XML tag: <....>
4535: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string);
4536: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4537: // finds the inner text between opening and closing tags
4538: function Easter(NYear: Integer): TDateTime;
4539: // returns the easter date of a year.

```

```

4540: function GetWeekNumber(Today: TDateTime): string;
4541: //gets a datecode. Returns year and weeknumber in format: YYWW
4542: function ParseNumber(const S: string): Integer;
4543: // parse number returns the last position, starting from 1
4544: function ParseDate(const S: string): Integer;
4545: // parse a SQL style date string from positions 1,
4546: // starts and ends with #
4547:
4548: *****unit JvJCLUtils;*****
4549: PathDelim', 'String').SetString( '' );
4550: DriveDelim', 'String').SetString( ':' );
4551: PathSep', 'String').SetString( ';' );
4552: AllFilesMask', 'String').SetString( '*' );
4553: PathDelim', 'String').SetString( '/' );
4554: AllFilesMask', 'String').SetString( '*' );
4555: NullHandle', 'LongInt').SetInt( 0 );
4556: USDecimalSeparator', 'String').SetString( '.' );
4557: CL.AddClassN(CL.FindClass('TOBJECT'), 'EJVConvertError');
4558: CL.AddTypeS('TFileTime', 'Integer');
4559: // CL.AddTypeS('TFormatSettings', 'record DecimalSeparator : Char; end');
4560: Function SendRectMessage( Handle : THandle; Msg : Integer; wParam : longint; var R : TRect ) : Integer';
4561: //CL.AddDelphiFunction('Function SendStructMessage( Handle : THandle; Msg : Integer; wParam : WPARAM; var Data ) : Integer');
4562: Function ReadCharsFromStream(Stream: TStream; var Buf:array of PChar;BufSize:Integer):Integer';
4563: Function WriteStringToStream(Stream: TStream; const Buf: AnsiString;BufSize:Integer): Integer';
4564: Function UTF8ToString( const S : UTF8String ) : string';
4565: //CL.AddConstantN('DefaultDateOrder','').SetString('doDMY');
4566: CL.AddConstantN('CenturyOffset','Byte').SetUInt( 60 );
4567:
4568: function VarIsInt(Value: Variant): Boolean;
4569: // VarIsInt returns VarIsOrdinal-[varBoolean]
4570: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4571: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4572: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4573: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4574: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4575: function GetWordOnPos(const S: string; const P: Integer): string;
4576: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4577: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4578: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4579: { GetWordOnPosEx working like GetWordOnPos function, but
4580:   also returns Word position in iBeg, iEnd variables }
4581: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4582: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4583: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4584: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4585: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4586: { GetEndPosCaret returns the caret position of the last char. For the position
4587:   after the last char of Text you must add 1 to the returned X value. }
4588: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4589: { GetEndPosCaret returns the caret position of the last char. For the position
4590:   after the last char of Text you must add 1 to the returned X value. }
4591: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4592: function SubStrBySeparator(const S:string;const Index:Integer;const
        Separator:string;StartIndex:Int=1):string;
4593: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
        Separator:WideString;StartIndex:Int:WideString;
4594: { SubStrEnd same to previous function but Index numerated from the end of string }
4595: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4596: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4597: function SubWord(P: PChar; var P2: PChar): string;
4598: function CurrencyByWord(Value: Currency): string;
4599: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4600: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4601: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4602: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4603: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4604: { ReplaceString searches for all substrings, OldPattern,
4605:   in a string, S, and replaces them with NewPattern }
4606: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4607: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
        WideString;StartIndex:Integer=1):WideString;
4608: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4609: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
        SUPPORTS_INLINE}
4610: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4611: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
        SUPPORTS_INLINE}
4612:
4613: { Next 4 function for russian chars transliterating.
4614:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4615: procedure Dos2Win(var S: AnsiString);
4616: procedure Win2Dos(var S: AnsiString);
4617: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4618: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4619: function Win2Koi(const S: AnsiString): AnsiString;
4620: { FillString fills the string Buffer with Count Chars }
4621: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4622: procedure FillString1(var Buffer: string; StartIndex,Count:Integer; const Value: Char); overload;

```

```

4623: { MoveString copies Count Chars from Source to Dest }
4624: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
4625: inline; {$ENDIF SUPPORTS_INLINE} overload;
4626: procedure MoveString1(const Source: string; SrcStartIdx: Integer; var Dest: string;
4627: DstStartIdx: Integer; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4628: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4629: procedure FillWideChar(var Buffer: Buffer; Count: Integer; const Value: WideChar);
4630: { MoveWideChar copies Count WideChars from Source to Dest }
4631: procedure MoveWideChar(const Source: var Dest; Count: Integer); {$IFDEF SUPPORTS_INLINE}inline; {$ENDIF
4632: SUPPORTS_INLINE}
4633: { FillNativeChar fills Buffer with Count NativeChars }
4634: procedure FillNativeChar(var Buffer: Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
4635: SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4636: { MoveWideChar copies Count WideChars from Source to Dest }
4637: procedure MoveNativeChar(const Source: var Dest; Count: Integer); // D2009 internal error {$IFDEF
4638: SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4639: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4640: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4641: { Spaces returns string consists on N space chars }
4642: function Spaces(const N: Integer): string;
4643: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4644: function AddSpaces(const S: string; const N: Integer): string;
4645: function SpacesW(const N: Integer): WideString;
4646: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4647: { function LastDateRUS for russian users only }
4648: { returns date relative to current date: 'äà àïÿ iàçàä' }
4649: function LastDateRUS(const Dat: TDateTime): string;
4650: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4651: function CurrencyToStr(const Cur: Currency): string;
4652: { HasChar returns True, if Char, Ch, contains in string, S }
4653: function HasChar(const Ch: Char; const S: string): Boolean;
4654: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$IFDEF SUPPORTS_INLINE}
4655: function HasAnyChar(const Chars: string; const S: string): Boolean;
4656: { $IFNDEF COMPILER12_UP }
4657: function CharInSet(const Ch: AnsiChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4658: { $ENDIF ~COMPILER12_UP }
4659: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF
4660: SUPPORTS_INLINE}
4661: function CountOfChar(const Ch: Char; const S: string): Integer;
4662: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4663: SUPPORTS_INLINE}
4664: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4665: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4666: function StrPosW(S, SubStr: PWideChar): PWideChar;
4667: function StrLenW(S: PWideChar): Integer;
4668: function TrimW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4669: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
4670: SUPPORTS_INLINE}
4671: function TrimRightW(const S: WideString): WideString; inline; {$IFDEF SUPPORTS_INLINE}
4672: TPixelFormat', '( pfDevice, pflbit, pf4bit, pf8bit, pf24bit )
4673: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4674: Function GetBitmapPixelFormat(Bitmap : TBitmap) : TPixelFormat
4675: Function GetPaletteBitmapFormat(Bitmap : TBitmap) : TPixelFormat
4676: Procedure SetBitmapPixelFormat(Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4677: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4678: Procedure GrayscaleBitmap(Bitmap : TBitmap)
4679: Function BitmapToMemory(Bitmap : TBitmap; Colors : Integer) : TStream
4680: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4681: Function ScreenPixelFormat : TPixelFormat
4682: Function ScreenColorCount : Integer
4683: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4684: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4685: // SIRegister_TJvGradient(CL);
4686: {***** files routines}
4687: procedure SetDelimitedText(List: TStringList; const Text: string; Delimiter: Char);
4688: const
4689: { $IFDEF MSWINDOWS}
4690: DefaultCaseSensitivity = False;
4691: {$ENDIF MSWINDOWS}
4692: { $IFDEF UNIX}
4693: DefaultCaseSensitivity = True;
4694: {$ENDIF UNIX}
4695: { GenTempFileName returns temporary file name on
4696: drive, there FileName is placed }
4697: function GenTempFileName(FileName: string): string;
4698: { GenTempFileNameExt same to previous function, but
4699: returning filename has given extension, FileExt }
4700: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4701: { ClearDir clears folder Dir }
4702: function ClearDir(const Dir: string): Boolean;
4703: { DeleteDir clears and then delete folder Dir }
4704: function DeleteDir(const Dir: string): Boolean;
4705: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4706: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4707: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4708: Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4709: function FileEquMasks(FileName, Masks: TFileName;
4710: CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;

```

```

4705: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4706: {$IFDEF MSWINDOWS}
4707: { LZFileExpand expand file, FileSource,
4708:   into FileDest. Given file must be compressed, using MS Compress program }
4709: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4710: {$ENDIF MSWINDOWS}
4711: { FileGetInfo fills SearchRec record for specified file attributes}
4712: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4713: { HasSubFolder returns True, if folder APath contains other folders }
4714: function HasSubFolder(APath: TFileName): Boolean;
4715: { IsEmptyFolder returns True, if there are no files or
4716:   folders in given folder, APath}
4717: function IsEmptyFolder(APath: TFileName): Boolean;
4718: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4719: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4720: { AddPath returns FileName with Path, if FileName not contain any path }
4721: function AddPath(const FileName, Path: TFileName): TFileName;
4722: function AddPaths(const PathList, Path: string): string;
4723: function ParentPath(const Path: TFileName): TFileName;
4724: function FindInPath(const FileName, PathList: string): TFileName;
4725: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4726: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4727: { HasParam returns True, if program running with specified parameter, Param }
4728: function HasParam(const Param: string): Boolean;
4729: function HasSwitch(const Param: string): Boolean;
4730: function Switch(const Param: string): string;
4731: { ExePath returns ExtractFilePath(ParamStr(0)) }
4732: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4733: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4734: //function FileTimeToDate(FT: TFileTime): TDateTime;
4735: procedure FileTimeToDosDateTime(FT: TFileTime; out Dft: DWORD);
4736: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4737: {**** Graphic routines }
4738: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4739: function IsTTFontSelected(const DC: HDC): Boolean;
4740: function KeyPressed(VK: Integer): Boolean;
4741: Function isKeyPressed: boolean; //true if key on memo2 (shell output) is pressed
4742: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4743: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4744: {**** Color routines }
4745: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4746: function RGBToBGR(Value: Cardinal): Cardinal;
4747: //function ColorToPrettyName(Value: TColor): string;
4748: //function PrettyNameToColor(const Value: string): TColor;
4749: {**** other routines }
4750: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4751: function IntPower(Base, Exponent: Integer): Integer;
4752: function ChangeTopException(E: TObject); // Linux version writes error message to ErrOutput
4753: function StrToBool(const S: string): Boolean;
4754: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4755: function VarToInt(V: Variant): Integer;
4756: function VarToFloat(V: Variant): Double;
4757: { following functions are not documented because they not work properly sometimes, so do not use them }
4758: // (rom) ReplaceString1, GetSubStr removed
4759: function GetLongFileName(const FileName: string): string;
4760: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4761: function GetParameter: string;
4762: function GetComputerID: string;
4763: function GetComputerName: string;
4764: {**** string routines }
4765: { ReplaceAllStrings searches for all substrings, Words,
4766:   in a string, S, and replaces them with Frases with the same Index. }
4767: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4768: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4769:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with the
4770:   same Index, and then update NewSelStart variable }
4771: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4772: { CountOfLines calculates the lines count in a string, S,
4773:   each line must be separated from another with CrLf sequence }
4774: function CountOfLines(const S: string): Integer;
4775: { DeleteLines deletes all lines from strings which in the words, words.
4776:   The word of will be deleted from strings. }
4777: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4778: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4779:   Lines contained only spaces also deletes. }
4780: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4781: { SQLAddWhere addes or modifies existing where-statement, where,
4782:   to the strings, SQL. Note: If strings SQL already contains where-statement,
4783:   it must be started on the begining of any line }
4784: {**** files routines - }
4785: {$IFDEF MSWINDOWS}
4786: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4787:   Resource can be compressed using MS Compress program}
4788: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4789: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
4790: Boolean;
4791: {$ENDIF MSWINDOWS}

```

```

4792: { IniReadSection read section, Section, from ini-file,
4793:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4794:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.)
4795: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4796: { LoadTextFile load text file, FileName, into string }
4797: function LoadTextFile(const FileName: TFileName): string;
4798: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4799: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4800: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4801: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4802: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4803: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4804: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4805: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S:string;const CalcHeight:Boolean):Integer;
4806: { RATextCalcHeight calculate needed height for
4807:   correct output, using RATextOut or RATextOutEx functions }
4808: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4809: { Cinema draws some visual effect }
4810: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4811: { Roughed fills rect with special 3D pattern }
4812: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4813: { 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 }
4814: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4815: { TextWidth calculate text with for writing using standard desktop font }
4816: function TextWidth(const AStr: string): Integer;
4817: { TextHeight calculate text height for writing using standard desktop font }
4818: function TextHeight(const AStr: string): Integer;
4819: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4820: procedure Error(const Msg: string);
4821: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4822:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4823: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4824: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4825:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4826: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4827:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4828: function ItemHtPlain(const Text: string): string;
4829: { ClearList - clears list of TObject }
4830: procedure ClearList(List: TList);
4831: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
4832: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4833: { RTTI support }
4834: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4835: function GetPropStr(Obj: TObject; const PropName: string): string;
4836: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4837: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4838: procedure PrepareIniSection(Ss: TStrings);
4839: { following functions are not documented because they are don't work properly, so don't use them }
4840: // (rom) from JvBandWindows to make it obsolete
4841: function PointL(const X, Y: Longint): TPoint; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4842: // (rom) from JvBandUtils to make it obsolete
4843: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4844: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4845: function CreateIconFromClipboard: TIcon;
4846: { begin JVIconClipboardUtils } { Icon clipboard routines }
4847: function CF_ICON: Word;
4848: procedure AssignClipboardIcon(Icon: TIcon);
4849: { Real-size icons support routines (32-bit only) }
4850: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4851: function CreateRealSizeIcon(Icon: TIcon): HICON;
4852: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4853: {end JVIconClipboardUtils }
4854: function CreateScreenCompatibleDC: HDC;
4855: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF}
4856: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4857: { begin JvRLE } // (rom) changed API for inclusion in JCL
4858: procedure RleCompressTo(InStream, OutStream: TStream);
4859: procedure RleDecompressTo(InStream, OutStream: TStream);
4860: procedure RleCompress(Stream: TStream);
4861: procedure RleDecompress(Stream: TStream);
4862: { end JvRLE } { begin JvDateUtil }
4863: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4864: function IsLeapYear(AYear: Integer): Boolean;
4865: function DaysInAMonth(const AYear, AMonth: Word): Word;
4866: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4867: function FirstDayOfPrevMonth: TDateTime;
4868: function LastDayOfPrevMonth: TDateTime;
4869: function FirstDayOfNextMonth: TDateTime;
4870: function ExtractDay(ADate: TDateTime): Word;
4871: function ExtractMonth(ADate: TDateTime): Word;
4872: function ExtractYear(ADate: TDateTime): Word;
4873: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4874: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime; inline; {$IFDEF SUPPORTS_INLINE}
4875: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4876: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4877: function ValidDate(ADate: TDateTime): Boolean;
4878: procedure DateDiff(DATE1, DATE2: TDateTime; var Days, Months, Years: Word);

```

```

4879: function MonthsBetween(Date1, Date2: TDateTime): Double;
4880: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4881: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4882: function DaysBetween(Date1, Date2: TDateTime): Longint;
4883: { The same as previous but if Date2 < Date1 result = 0 }
4884: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4885: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4886: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4887: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4888: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4889: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4890: { String to date conversions }
4891: function GetDateOrder(const DateFormat: string): TDateOrder;
4892: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4893: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4894: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4895: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4896: //function DefDateFormat(AFourDigitYear: Boolean): string;
4897: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4898: function FormatLongDate(Value: TDateTime): string;
4899: function FormatLongDateTime(Value: TDateTime): string;
4900: { end JVDateUtil }
4901: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4902: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4903: { begin JvStrUtils } { ** Common string handling routines ** }
4904: {$IFDEF UNIX}
4905: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4906: const ToCode, FromCode: AnsiString): Boolean;
4907: function iconvstring(const S, ToCode, FromCode: AnsiString): string;
4908: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4909: function OemStrToAnsi(const S: AnsiString): AnsiString;
4910: function AnsiStrToOem(const S: AnsiString): AnsiString;
4911: {$ENDIF UNIX}
4912: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4913: { StrToOem translates a string from the Windows character set into the OEM character set. }
4914: function OemToStr(const OemStr: AnsiString): AnsiString;
4915: { OemToStr translates a string from the OEM character set into the Windows character set. }
4916: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4917: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4918: function ReplaceStr(const S, Srch, Replace: string): string;
4919: { Returns string with every occurrence of Srch string replaced with Replace string. }
4920: function DelSpace(const S: string): string;
4921: { DelSpace return a string with all white spaces removed. }
4922: function DelChars(const S: string; Chr: Char): string;
4923: { DelChars return a string with all Chr characters removed. }
4924: function DelBSpace(const S: string): string;
4925: { DelBSpace trims leading spaces from the given string. }
4926: function DelEspace(const S: string): string;
4927: { DelEspace trims trailing spaces from the given string. }
4928: function DelRSpace(const S: string): string;
4929: { DelRSpace trims leading and trailing spaces from the given string. }
4930: function DelSpace1(const S: string): string;
4931: { DelSpace1 return a string with all non-single white spaces removed. }
4932: function Tab2Space(const S: string; Numb: Byte): string;
4933: { Tab2Space converts any tabulation character in the given string to the
4934: Numb spaces characters. }
4935: function NPos(const C: string; S: string; N: Integer): Integer;
4936: { NPos searches for a N-th position of substring C in a given string. }
4937: function MakeStr(C: Char; N: Integer): string; overload;
4938: {$IFDEF COMPILER12_UP}
4939: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4940: {$ENDIF !COMPILER12_UP}
4941: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4942: { MakeStr return a string of length N filled with character C. }
4943: function AddChar(C: Char; const S: string; N: Integer): string;
4944: { AddChar return a string left-padded to length N with characters C. }
4945: function AddCharR(C: Char; const S: string; N: Integer): string;
4946: { AddCharR return a string right-padded to length N with characters C. }
4947: function LeftStr(const S: string; N: Integer): string;
4948: { LeftStr return a string right-padded to length N with blanks. }
4949: function RightStr(const S: string; N: Integer): string;
4950: { RightStr return a string left-padded to length N with blanks. }
4951: function CenterStr(const S: string; Len: Integer): string;
4952: { CenterStr centers the characters in the string based upon the Len specified. }
4953: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4954: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4955: -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4956: function CompText(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4957: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4958: function Copy2Symb(const S: string; Symb: Char): string;
4959: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4960: function Copy2SymbDel(var S: string; Symb: Char): string;
4961: { Copy2SymbDel returns a substring of a string S from beginning to first
4962: character Symb and removes this substring from S. }
4963: function Copy2Space(const S: string): string;
4964: { Copy2Space returns a substring of a string S from beginning to first white space. }
4965: function Copy2SpaceDel(var S: string): string;
4966: { Copy2SpaceDel returns a substring of a string S from beginning to first
4967: white space and removes this substring from S. }

```

```

4968: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4969: { Returns string, with the first letter of each word in uppercase,
4970:   all other letters in lowercase. Words are delimited by WordDelims. }
4971: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4972: { WordCount given a set of word delimiters, returns number of words in S. }
4973: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4974: { Given a set of word delimiters, returns start position of N'th word in S. }
4975: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4976: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TSysCharSet; var Pos: Integer): string;
4977: function ExtractDelimited(N: Integer; const S: string; const Delims: TSysCharSet): string;
4978: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4979:   delimiters, return the N'th word in S. }
4980: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TSysCharSet): string;
4981: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4982:   that started from position Pos. }
4983: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4984: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4985: function QuotedString(const S: string; Quote: Char): string;
4986: { QuotedString returns the given string as a quoted string, using provided Quote character. }
4987: function ExtractQuotedString(const S: string; Quote: Char): string;
4988: { ExtractQuotedString removes the Quote characters from the beginning and
4989:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character. }
4990: function FindPart(const HelpWilds, InputStr: string): Integer;
4991: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4992: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4993: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4994: function XorString(const Key, Src: ShortString): ShortString;
4995: function XorEncode(const Key, Source: string): string;
4996: function XorDecode(const Key, Source: string): string;
4997: { ** Command line routines ** }
4998: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4999: { ** Numeric string handling routines ** }
5000: function Numb2USA(const S: string): string;
5001: { Numb2USA converts numeric string S to USA-format. }
5002: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
5003: { Dec2Hex converts the given value to a hexadecimal string representation
5004:   with the minimum number of digits (A) specified. }
5005: function Hex2Dec(const S: string): Longint;
5006: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
5007: function Dec2Numb(N: Int64; A, B: Byte): string;
5008: { Dec2Numb converts the given value to a string representation with the
5009:   base equal to B and with the minimum number of digits (A) specified. }
5010: function Numb2Dec(S: string; B: Byte): Int64;
5011: { Numb2Dec converts the given B-based numeric string to the corresponding
5012:   integer value. }
5013: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
5014: { IntToBin converts the given value to a binary string representation
5015:   with the minimum number of digits specified. }
5016: function IntToRoman(Value: Longint): string;
5017: { IntToRoman converts the given value to a roman numeric string representation. }
5018: function RomanToInt(const S: string): Longint;
5019: { RomanToInt converts the given string to an integer value. If the string
5020:   doesn't contain a valid roman numeric value, the 0 value is returned. }
5021: function FindNotBlankCharPos(const S: string): Integer;
5022: function FindNotBlankCharPosW(const S: WideString): Integer;
5023: function AnsiChangeCase(const S: string): string;
5024: function WideChangeCase(const S: string): string;
5025: function StartsText(const Substr, S: string): Boolean;
5026: function EndsText(const Substr, S: string): Boolean;
5027: function DequotedStr(const S: string; QuoteChar: Char = ''): string;
5028: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
5029: {end JvStringUtil}
5030: {$IFDEF UNIX}
5031: function GetTempFileName(const Prefix: AnsiString): AnsiString;
5032: {$ENDIF UNIX}
5033: { begin JvFileUtil }
5034: function FileDateTime(const FileName: string): TDateTime;
5035: function HasAttr(const FileName: string; Attr: Integer): Boolean;
5036: function DeleteFilesEx(const FileMasks: array of string): Boolean;
5037: function NormalDir(const DirName: string): string;
5038: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
5039: function ValidFileName(const FileName: string): Boolean;
5040: {$IFDEF MSWINDOWS}
5041: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5042: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5043: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5044: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5045: {$ENDIF MSWINDOWS}
5046: function GetWindowsDir: string;
5047: function GetSystemDir: string;
5048: function ShortToLongFileName(const ShortName: string): string;
5049: function LongToShortFileName(const LongName: string): string;
5050: function ShortToLongPath(const ShortName: string): string;
5051: function LongToShortPath(const LongName: string): string;
5052: {$IFDEF MSWINDOWS}
5053: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
5054: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5055: {$ENDIF MSWINDOWS}
5056: { end JvFileUtil }

```

```

5057: // Works like PtInRect but includes all edges in comparision
5058: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5059: // Works like PtInRect but excludes all edges from comparision
5060: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5061: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5062: function IsFourDigitYear: Boolean;
5063: { moved from JVJVCLUtils }
5064: //Open an object with the shell (url or something like that)
5065: function OpenObject(const Value: string): Boolean; overload;
5066: function OpenObject(Value: PChar): Boolean; overload;
5067: {$IFDEF MSWINDOWS}
5068: //Raise the last Exception
5069: procedure RaiseLastWin32; overload;
5070: procedure RaiseLastWin32(const Text: string); overload;
5071: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
5072: function GetFileVersion(const AFileName: string): Cardinal;
5073: {$EXTERNALSYM GetFileVersion}
5074: //Get version of Shell.dll
5075: function GetShellVersion: Cardinal;
5076: {$EXTERNALSYM GetShellVersion}
5077: // CD functions on HW
5078: procedure OpenCdDrive;
5079: procedure CloseCdDrive;
5080: // returns True if Drive is accessible
5081: function DiskInDrive(Drive: Char): Boolean;
5082: {$IFDEF MSWINDOWS}
5083: //Same as linux function ;
5084: procedure PError(const Text: string);
5085: // execute a program without waiting
5086: procedure Exec(const FileName, Parameters, Directory: string);
5087: // execute a program and wait for it to finish
5088: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5089: // returns True if this is the first instance of the program that is running
5090: function FirstInstance(const ATitle: string): Boolean;
5091: // restores a window based on it's classname and Caption. Either can be left empty
5092: // to widen the search
5093: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5094: // manipulate the traybar and start button
5095: procedure HideTraybar;
5096: procedure ShowTraybar;
5097: procedure ShowStartButton(Visible: Boolean = True);
5098: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5099: procedure MonitorOn;
5100: procedure MonitorOff;
5101: procedure LowPower;
5102: // send a key to the window named AppName
5103: function SendKey(const AppName: string; Key: Char): Boolean;
5104: {$IFDEF MSWINDOWS}
5105: // returns a list of all win currently visible, the Objects property is filled with their window handle
5106: procedure GetVisibleWindows(List: TStrings);
5107: Function GetVisibleWindowsF( List : TStrings):TStrings';
5108: // associates an extension to a specific program
5109: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5110: procedure AddToRecentDocs(const FileName: string);
5111: function GetRecentDocs: TStringList;
5112: {$IFDEF MSWINDOWS}
5113: function CharIsMoney(const Ch: Char): Boolean;
5114: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5115: function IntToExtended(I: Integer): Extended;
5116: { GetChangedText works out the new text given the current cursor pos & the key pressed
      It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5117: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5118: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5119: function StrIsInteger(const S: string): Boolean;
5120: //function StrIsFloatMoney(const Ps: string): Boolean;
5121: function StrIsFloatMoney(const Ps: string): Boolean;
5122: function StrIsDateTime(const Ps: string): Boolean;
5123: function PreformatDateString(Ps: string): string;
5124: function BooleanToInteger(const B: Boolean): Integer;
5125: function StringToBoolean(const Ps: string): Boolean;
5126: function SafeStrToDate(const Ps: string): TDateTime;
5127: function SafeStrToDate(const Ps: string): TDateTime;
5128: function SafeStrToTime(const Ps: string): TDateTime;
5129: function StrDelete(const psSub, psMain: string): string;
5130: { returns the fractional value of pcValue}
5131: function TimeOnly(pcValue: TDateTime): TTime;
5132: { returns the integral value of pcValue }
5133: function DateOnly(pcValue: TDateTime): TDate;
5134: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5135: const { TDateTime value used to signify Null value}
5136: NullEquivalentDate: TDateTime = 0.0;
5137: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5138: // Replacement for Win32Check to avoid platform specific warnings in D6
5139: function OSCheckRetVal: Boolean;
5140: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
      Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
      not be forced to use FileCtrl unnecessarily }
5141: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;

```

```

5144: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5145: { MinimizeString truncates long string, S, and appends...'symbols,if Length of S is more than MaxLen }
5146: function MinimizeString(const S: string; const MaxLen: Integer): string;
5147: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=SW_SHOWDEFAULT);
5148: { 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.}
5149: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5150: {$ENDIF MSWINDOWS}
5151: procedure ResourceNotFound(ResID: PChar);
5152: function EmptyRect: TRect;
5153: function RectWidth(R: TRect): Integer;
5154: function RectHeight(R: TRect): Integer;
5155: function CompareRect(const R1, R2: TRect): Boolean;
5156: procedure RectNormalize(var R: TRect);
5157: function RectIsSquare(const R: TRect): Boolean;
5158: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5159: //If AMaxSize = -1 ,then auto calc Square's max size
5160: {$IFDEF MSWINDOWS}
5161: procedure FreeUnusedOle;
5162: function GetWindowsVersion: string;
5163: function LoadDLL(const LibName: string): THandle;
5164: function RegisterServer(const ModuleName: string): Boolean;
5165: function UnregisterServer(const ModuleName: string): Boolean;
5166: {$ENDIF MSWINDOWS}
5167: { String routines }
5168: function GetEnvVar(const VarName: string): string;
5169: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5170: function StringToPChar(var S: string): PChar;
5171: function StrPAlloc(const S: string): PChar;
5172: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5173: function DropT(const S: string): string;
5174: { Memory routines }
5175: function AllocMemo(Size: Longint): Pointer;
5176: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5177: procedure FreeMemo(var fpBlock: Pointer);
5178: function GetMemoSize(fpBlock: Pointer): Longint;
5179: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5180: { Manipulate huge pointers routines }
5181: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5182: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5183: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5184: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5185: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5186: function WindowClassName(Wnd: THandle): string;
5187: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5188: procedure ActivateWindow(Wnd: THandle);
5189: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5190: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5191: { SetWindowTop put window to top without recreating window }
5192: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5193: procedure CenterWindow(Wnd: THandle);
5194: function MakeVariant(const Values: array of Variant): Variant;
5195: { Convert dialog units to pixels and backwards }
5196: {$IFDEF MSWINDOWS}
5197: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5198: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5199: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5200: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5201: {$ENDIF MSWINDOWS}
5202: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5203: {$IFDEF BCB}
5204: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5205: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5206: {$ELSE}
5207: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5208: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5209: {$ENDIF BCB}
5210: {$IFDEF MSWINDOWS}
5211: { BrowseForFolderNative displays Browse For Folder dialog }
5212: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5213: {$ENDIF MSWINDOWS}
5214: procedure AntiAlias(Clip: TBitmap);
5215: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin, XFinal, YFinal: Integer);
5216: procedure CopyRecttDBits(ACanvas: TCanvas; const DestRect: TRect;
5217:   ABitmap: TBitmap; const SourceRect: TRect);
5218: function IsTrueType(const FontName: string): Boolean;
5219: // Removes all non-numeric characters from AValue and returns the resulting string
5220: function TextToValText(const AValue: string): string;
5221: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5222: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5223: Function ReplaceRegExpr(const ARegExpr,AInputStr,
      AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5224: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5225: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5226:
5227: *****unit uPSI_JvTFUtils;
5228: Function JExtractYear( ADate : TDateTime) : Word

```

```

5229: Function JExtractMonth( ADate : TDateTime ) : Word
5230: Function JExtractDay( ADate : TDateTime ) : Word
5231: Function ExtractHours( ATime : TDateTime ) : Word
5232: Function ExtractMins( ATime : TDateTime ) : Word
5233: Function ExtractSecs( ATime : TDateTime ) : Word
5234: Function ExtractMSecs( ATime : TDateTime ) : Word
5235: Function FirstOfMonth( ADate : TDateTime ) : TDateTime
5236: Function GetDayOfNthDOW( Year, Month, DOW, N : Word ) : Word
5237: Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer ) : Word
5238: Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer )
5239: Procedure IncDOW( var DOW : TTFFDayOfWeek; N : Integer )
5240: Procedure IncDays( var ADate : TDateTime; N : Integer )
5241: Procedure IncWeeks( var ADate : TDateTime; N : Integer )
5242: Procedure IncMonths( var ADate : TDateTime; N : Integer )
5243: Procedure IncYears( var ADate : TDateTime; N : Integer )
5244: Function EndOfMonth( ADate : TDateTime ) : TDateTime
5245: Function IsFirstOfMonth( ADate : TDateTime ) : Boolean
5246: Function IsEndOfMonth( ADate : TDateTime ) : Boolean
5247: Procedure EnsureMonth( Month : Word )
5248: Procedure EnsureDOW( DOW : Word )
5249: Function EqualDates( D1, D2 : TDateTime ) : Boolean
5250: Function Lesser( N1, N2 : Integer ) : Integer
5251: Function Greater( N1, N2 : Integer ) : Integer
5252: Function GetDivLength( TotalLength, DivCount, DivNum : Integer ) : Integer
5253: Function GetDivNum( TotalLength, DivCount, X : Integer ) : Integer
5254: Function GetDivStart( TotalLength, DivCount, DivNum : Integer ) : Integer
5255: Function DOWToBorl( ADOW : TTFFDayOfWeek ) : Integer
5256: Function BorlToDOW( BorlDOW : Integer ) : TTFFDayOfWeek
5257: Function DateToDOW( ADate : TDateTime ) : TTFFDayOfWeek
5258: Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
  AFont:TFont; AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string )
5259: Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
  HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string )
5260: Function JRectWidth( ARect : TRect ) : Integer
5261: Function JRectHeight( ARect : TRect ) : Integer
5262: Function JEmptyRect : TRect
5263: Function IsClassByName( Obj : TObject; ClassName : string ) : Boolean
5264:
5265: procedure SIRегистre_MSysUtils(CL: TPSPascalCompiler);
5266: begin
5267:   Procedure HideTaskBarButton( hWindow : HWND )
5268:   Function msLoadStr( ID : Integer ) : String
5269:   Function msFormat( fmt : String; params : array of const ) : String
5270:   Function msFileExists( const FileName : String ) : Boolean
5271:   Function msIntToStr( Int : Int64 ) : String
5272:   Function msStrPas( const Str : PChar ) : String
5273:   Function msRenameFile( const OldName, NewName : String ) : Boolean
5274:   Function CutFileName( s : String ) : String
5275:   Function GetVersionInfo( var VersionString : String ) : DWORD
5276:   Function FormatTime( t : Cardinal ) : String
5277:   Function msCreateDir( const Dir : string ) : Boolean
5278:   Function SetAutoRun( NeedAutoRun : Boolean; AppName : String ) : Boolean
5279:   Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword ) : DWORD
5280:   Function msStrLen( Str : PChar ) : Integer
5281:   Function msDirectoryExists( const Directory : String ) : Boolean
5282:   Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String ) : String
5283:   Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte ) : LongBool
5284:   Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
5285:   Procedure SetEditWndProc( hWnd : HWND; ptr : TObject )
5286:   Function GetTextFromFile( filename : String ) : string
5287:   Function IsTopMost( hWnd : HWND ) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002 );
5288:   Function msStrToIntDef( const s : String; const i : Integer ) : Integer
5289:   Function msStrToInt( s : String ) : Integer
5290:   Function GetItemText( hDlg : THandle; ID : DWORD ) : String
5291: end;
5292:
5293: procedure SIRегистre_ESBMaths2(CL: TPSPascalCompiler);
5294: begin
5295:   //TDynFloatArray', 'array of Extended
5296:   TDynWordArray', 'array of LongWord
5297:   TDynIntArray', 'array of LongInt
5298:   TDynFloatMatrix', 'array of TDynFloatArray
5299:   TDynWordMatrix', 'array of TDynWordArray
5300:   TDynIntMatrix', 'array of TDynIntArray
5301:   Function SquareAll( const X : TDynFloatArray ) : TDynFloatArray
5302:   Function InverseAll( const X : TDynFloatArray ) : TDynFloatArray
5303:   Function LnAll( const X : TDynFloatArray ) : TDynFloatArray
5304:   Function Log10All( const X : TDynFloatArray ) : TDynFloatArray
5305:   Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended ) : TDynFloatArray
5306:   Function AddVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5307:   Function SubVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5308:   Function MultVectors( const X, Y : TDynFloatArray ) : TDynFloatArray
5309:   Function DotProduct( const X, Y : TDynFloatArray ) : Extended
5310:   Function MNorm( const X : TDynFloatArray ) : Extended
5311:   Function MatrixIsRectangular( const X : TDynFloatMatrix ) : Boolean
5312:   Procedure MatrixDimensions( const X : TDynFloatMatrix; var Rows, Columns : LongWord; var Rectangular : Boolean );
5313:   Function MatrixIsSquare( const X : TDynFloatMatrix ) : Boolean
5314:   Function MatricesSameDimensions( const X, Y : TDynFloatMatrix ) : Boolean
5315:   Function AddMatrices( const X, Y : TDynFloatMatrix ) : TDynFloatMatrix

```

```

5316: Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5317: Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5318: Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5319: Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5320: Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5321: Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5322: Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5323: Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5324: end;
5325:
5326: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5327: begin
5328:   'ESBMinSingle','Single').setExtended( 1.5e-45);
5329:   'ESBMaxSingle','Single').setExtended( 3.4e+38);
5330:   'ESBMinDouble','Double').setExtended( 5.0e-324);
5331:   'ESBMaxDouble','Double').setExtended( 1.7e+308);
5332:   'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5333:   'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5334:   'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5335:   'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5336:   'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5337:   'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5338:   'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5339:   'ESBSqrt10','Extended').setExtended( 3.162277660168379320);
5340:   'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5341:   'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5342:   'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5343:   'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5344:   'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5345:   'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5346:   'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5347:   'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5348:   'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5349:   'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5350:   'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5351:   'ESBe','Extended').setExtended( 2.7182818284590452354);
5352:   'ESBe2','Extended').setExtended( 7.3890560989306502272);
5353:   'ESBePi','Extended').setExtended( 23.140692632779269006);
5354:   'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5355:   'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5356:   'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5357:   'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5358:   'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5359:   'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5360:   'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5361:   'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5362:   'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5363:   'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5364:   'ESBPi','Extended').setExtended( 3.1415926535897932385);
5365:   'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5366:   'ESBTwopi','Extended').setExtended( 6.2831853071795864769);
5367:   'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5368:   'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5369:   'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5370:   'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5371:   'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5372:   'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5373:   'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5374:   'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5375:   'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5376:   'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5377:   'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5378:   'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5379:   'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5380:   'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5381:   'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5382:   //LongWord', 'Cardinal
5383:   TBitList', 'Word
5384:   Function UMul( const Num1, Num2 : LongWord) : LongWord
5385:   Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5386:   Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5387:   Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5388:   Function SameFloat( const X1, X2 : Extended) : Boolean
5389:   Function FloatIsZero( const X : Extended) : Boolean
5390:   Function FloatIsPositive( const X : Extended) : Boolean
5391:   Function FloatIsNegative( const X : Extended) : Boolean
5392:   Procedure IncLim( var B : Byte; const Limit : Byte)
5393:   Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5394:   Procedure IncLimW( var B : Word; const Limit : Word)
5395:   Procedure IncLimI( var B : Integer; const Limit : Integer)
5396:   Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5397:   Procedure DecLim( var B : Byte; const Limit : Byte)
5398:   Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5399:   Procedure DecLimW( var B : Word; const Limit : Word)
5400:   Procedure DecLimI( var B : Integer; const Limit : Integer)
5401:   Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5402:   Function MaxB( const B1, B2 : Byte) : Byte
5403:   Function MinB( const B1, B2 : Byte) : Byte
5404:   Function MaxSI( const B1, B2 : ShortInt) : ShortInt

```

```

5405: Function MinSI( const B1, B2 : ShortInt ) : ShortInt
5406: Function MaxW( const B1, B2 : Word ) : Word
5407: Function MinW( const B1, B2 : Word ) : Word
5408: Function esbMaxI( const B1, B2 : Integer ) : Integer
5409: Function esbMinI( const B1, B2 : Integer ) : Integer
5410: Function MaxL( const B1, B2 : LongInt ) : LongInt
5411: Function MinL( const B1, B2 : LongInt ) : LongInt
5412: Procedure SwapB( var B1, B2 : Byte )
5413: Procedure SwapSI( var B1, B2 : ShortInt )
5414: Procedure SwapW( var B1, B2 : Word )
5415: Procedure SwapI( var B1, B2 : SmallInt )
5416: Procedure SwapL( var B1, B2 : LongInt )
5417: Procedure SwapI32( var B1, B2 : Integer )
5418: Procedure SwapC( var B1, B2 : LongWord )
5419: Procedure SwapInt64( var X, Y : Int64 )
5420: Function esbSign( const B : LongInt ) : ShortInt
5421: Function Max4Word( const X1, X2, X3, X4 : Word ) : Word
5422: Function Min4Word( const X1, X2, X3, X4 : Word ) : Word
5423: Function Max3Word( const X1, X2, X3 : Word ) : Word
5424: Function Min3Word( const X1, X2, X3 : Word ) : Word
5425: Function MaxBArray( const B : array of Byte ) : Byte
5426: Function MaxWArray( const B : array of Word ) : Word
5427: Function MaxSIArray( const B : array of ShortInt ) : ShortInt
5428: Function MaxIArray( const B : array of Integer ) : Integer
5429: Function MaxLArray( const B : array of LongInt ) : LongInt
5430: Function MinBArray( const B : array of Byte ) : Byte
5431: Function MinWArray( const B : array of Word ) : Word
5432: Function MinSIArray( const B : array of ShortInt ) : ShortInt
5433: Function MinIArray( const B : array of Integer ) : Integer
5434: Function MinLArray( const B : array of Longint ) : LongInt
5435: Function SumBArray( const B : array of Byte ) : Byte
5436: Function SumBArray2( const B : array of Byte ) : Word
5437: Function SumSIArray( const B : array of ShortInt ) : ShortInt
5438: Function SumSIArray2( const B : array of ShortInt ) : Integer
5439: Function SumWArray( const B : array of Word ) : Word
5440: Function SumWArray2( const B : array of Word ) : LongInt
5441: Function SumIArray( const B : array of Integer ) : Integer
5442: Function SumLArray( const B : array of LongInt ) : LongInt
5443: Function SumLWArray( const B : array of LongWord ) : LongWord
5444: Function ESBDigits( const X : LongWord ) : Byte
5445: Function BitsHighest( const X : LongWord ) : Integer
5446: Function ESBBitsNeeded( const X : LongWord ) : Integer
5447: Function esbGCD( const X, Y : LongWord ) : LongWord
5448: Function esbLCM( const X, Y : LongInt ) : Int64
5449: //Function esbLCM( const X, Y : LongInt ) : LongInt
5450: Function RelativePrime( const X, Y : LongWord ) : Boolean
5451: Function Get87ControlWord : TBitList
5452: Procedure Set87ControlWord( const CWord : TBitList )
5453: Procedure SwapExt( var X, Y : Extended )
5454: Procedure SwapDbl( var X, Y : Double )
5455: Procedure SwapSing( var X, Y : Single )
5456: Function esbSgn( const X : Extended ) : ShortInt
5457: Function Distance( const X1, Y1, X2, Y2 : Extended ) : Extended
5458: Function ExtMod( const X, Y : Extended ) : Extended
5459: Function ExtRem( const X, Y : Extended ) : Extended
5460: Function CompMOD( const X, Y : Comp ) : Comp
5461: Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended )
5462: Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended )
5463: Function DMS2Extended( const Degs, Mins, Secs : Extended ) : Extended
5464: Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended )
5465: Function MaxExt( const X, Y : Extended ) : Extended
5466: Function MinExt( const X, Y : Extended ) : Extended
5467: Function MaxEArray( const B : array of Extended ) : Extended
5468: Function MinEArray( const B : array of Extended ) : Extended
5469: Function MaxSArray( const B : array of Single ) : Single
5470: Function MinSArray( const B : array of Single ) : Single
5471: Function MaxCarry( const B : array of Comp ) : Comp
5472: Function MinCarry( const B : array of Comp ) : Comp
5473: Function SumSArray( const B : array of Single ) : Single
5474: Function SumEArray( const B : array of Extended ) : Extended
5475: Function SumSqEArray( const B : array of Extended ) : Extended
5476: Function SumSgDiffBArray( const B : array of Extended; Diff : Extended ) : Extended
5477: Function SumXYEArray( const X, Y : array of Extended ) : Extended
5478: Function SumCArray( const B : array of Comp ) : Comp
5479: Function FactorialX( A : LongWord ) : Extended
5480: Function PermutationX( N, R : LongWord ) : Extended
5481: Function esbBinomialCoeff( N, R : LongWord ) : Extended
5482: Function IsPositiveEArray( const X : array of Extended ) : Boolean
5483: Function esbGeometricMean( const X : array of Extended ) : Extended
5484: Function esbHarmonicMean( const X : array of Extended ) : Extended
5485: Function ESBMean( const X : array of Extended ) : Extended
5486: Function esbSampleVariance( const X : array of Extended ) : Extended
5487: Function esbPopulationVariance( const X : array of Extended ) : Extended
5488: Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended )
5489: Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended )
5490: Function GetMedian( const SortedX : array of Extended ) : Extended
5491: Function GetMode( const SortedX : array of Extended; var Mode : Extended ) : Boolean
5492: Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended )
5493: Function ESBMagnitude( const X : Extended ) : Integer

```

```

5494: Function ESBTan( Angle : Extended ) : Extended
5495: Function ESBCot( Angle : Extended ) : Extended
5496: Function ESBcosec( const Angle : Extended ) : Extended
5497: Function ESBSec( const Angle : Extended ) : Extended
5498: Function ESBArcTan( X, Y : Extended ) : Extended
5499: Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5500: Function ESBArcCos( const X : Extended ) : Extended
5501: Function ESBArcSin( const X : Extended ) : Extended
5502: Function ESBArcSec( const X : Extended ) : Extended
5503: Function ESBArcCosec( const X : Extended ) : Extended
5504: Function ESBLog10( const X : Extended ) : Extended
5505: Function ESBLog2( const X : Extended ) : Extended
5506: Function ESBLogBase( const X, Base : Extended ) : Extended
5507: Function Pow2( const X : Extended ) : Extended
5508: Function IntPow( const Base : Extended; const Exponent : LongWord ) : Extended
5509: Function ESBIntPower( const X : Extended; const N : LongInt ) : Extended
5510: Function XtoY( const X, Y : Extended ) : Extended
5511: Function esbTenToY( const Y : Extended ) : Extended
5512: Function esbTwoToY( const Y : Extended ) : Extended
5513: Function LogXtoBaseY( const X, Y : Extended ) : Extended
5514: Function esbISqrt( const I : LongWord ) : Longword
5515: Function ILog2( const I : LongWord ) : LongWord
5516: Function IGreatestPowerOf2( const N : LongWord ) : LongWord
5517: Function ESBArCosh( X : Extended ) : Extended
5518: Function ESBArSinh( X : Extended ) : Extended
5519: Function ESBArTanh( X : Extended ) : Extended
5520: Function ESBcosh( X : Extended ) : Extended
5521: Function ESBsinh( X : Extended ) : Extended
5522: Function ESBtanh( X : Extended ) : Extended
5523: Function InverseGamma( const X : Extended ) : Extended
5524: Function esbGamma( const X : Extended ) : Extended
5525: Function esbLnGamma( const X : Extended ) : Extended
5526: Function esbBeta( const X, Y : Extended ) : Extended
5527: Function IncompleteBeta( X : Extended; P, Q : Extended ) : Extended
5528: end;
5529:
5530: ***** Integer Huge Cardinal Utils *****
5531: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean ) : uint64
5532: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean ) : uint32
5533: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean ) : uint64
5534: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean ) : uint32
5535: Function BitCount_8( Value : byte ) : integer
5536: Function BitCount_16( Value : uint16 ) : integer
5537: Function BitCount_32( Value : uint32 ) : integer
5538: Function BitCount_64( Value : uint64 ) : integer
5539: Function CountSetBits_64( Value : uint64 ) : integer TPrimalityTestNoticeProc',
5540: Procedure ( CountPrimalityTests : integer )
5541: Function gcd( a, b : THugeCardinal ) : THugeCardinal
5542: Function lcm( a, b : THugeCardinal ) : THugeCardinal
5543: Function isCoPrime( a, b : THugeCardinal ) : boolean
5544: Function isProbablyPrime( p: THugeCardinal; OnProgress: TProgress; var wasAborted: boolean): boolean
5545: Function hasSmallFactor( p : THugeCardinal ) : boolean
5546: //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer ) : boolean
5547: Function Inverse( Prime, Modulus : THugeCardinal ) : THugeCardinal
5548: Const('StandardExponent','LongInt'( 65537 );
5549: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5550: Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal ) : boolean'
5551:
5552: procedure SIRegister_xrtl_math_Integer(CL: TPSPPascalCompiler);
5553: begin
5554:   AddTypeS('TXRTLInteger', 'array of Integer
5555:   AddClassN(FindClass('TOBJECT'), 'EXRTLMathException
5556:   (FindClass('TOBJECT')),'EXRTLExtendInvalidArgument
5557:   AddClassN(FindClass('TOBJECT')),'EXRTLDivisionByZero
5558:   AddClassN(FindClass('TOBJECT')),'EXRTLExpInvalidArgument
5559:   AddClassN(FindClass('TOBJECT')),'EXRTLInvalidRadix
5560:   AddClassN(FindClass('TOBJECT')),'EXRTLInvalidRadixDigit
5561:   AddClassN(FindClass('TOBJECT')),'EXRTLRootInvalidArgument
5562:   'BitsPerByte','LongInt'( 8 );
5563:   BitsPerDigit,'LongInt'( 32 );
5564:   SignBitMask','LongWord( $80000000 );
5565:   Function XRTLAdjustBits( const ABits : Integer ) : Integer
5566:   Function XRTLlength( const AInteger : TXRTLInteger ) : Integer
5567:   Function XRTLDataBits( const AInteger : TXRTLInteger ) : Integer
5568:   Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5569:   Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5570:   Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5571:   Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer ) : Integer
5572:   Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer ) : Boolean
5573:   Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Int;Sign:Int;var AResult:TXRTLInteger):Int;
5574:   Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5575:   Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5576:   Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5577:   Procedure XRTLNNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5578:   Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)

```

```

5579: Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5580: Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5581: Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5582: Procedure XRTLZero( var AInteger : TXRTLInteger)
5583: Procedure XRTLOne( var AInteger : TXRTLInteger)
5584: Procedure XRTLMOne( var AInteger : TXRTLInteger)
5585: Procedure XRTLTwo( var AInteger : TXRTLInteger)
5586: Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5587: Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5588: Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5589: Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5590: Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Int;
5591: Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5592: Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Int;
5593: Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5594: Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5595: Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5596: Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Int
5597: Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5598: Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger, RResult : TXRTLInteger)
5599: Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5600: Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5601: Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5602: Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHHighApproxResult:TXRTLInteger)
5603: Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHHighApproxResult:TXRTLInteger);
5604: Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5605: Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult:TXRTLInteger)
5606: Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5607: Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5608: Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5609: Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5610: Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5611: Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5612: Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5613: Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5614: Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5615: Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5616: Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5617: Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5618: Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5619: Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5620: Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5621: Procedure XRTLFFromHex( const Value : string; var AResult : TXRTLInteger)
5622: Procedure XRTLFFromBin( const Value : string; var AResult : TXRTLInteger)
5623: Procedure XRTLFFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5624: Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5625: Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5626: Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5627: Procedure XRTLAppend( const ALow, AHHigh : TXRTLInteger; var AResult : TXRTLInteger)
5628: Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5629: Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5630: Procedure XRTLMInMax(const AInteger1, AInteger2:TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5631: Procedure XRTLMIn( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5632: Procedure XRTLMIn1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5633: Procedure XRTLMMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5634: Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5635: Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5636: Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5637: Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5638: Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5639: end;
5640:
5641: procedure SIRegister_JvXPCoreUtils(CL: TPSCompiler);
5642: begin
5643:   Function JvXPMETHODSEqual( const Method1, Method2 : TMethod) : Boolean
5644:   Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5645:   Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5646:   Procedure JvXPADJUSTBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5647:   Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLns;const
      AColor:TColor;Rect:TRect;
5648:   Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5649:   Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5650:   Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5651:   Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5652:   Procedure JvXPSetDrawFlags(const AAlignment:TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5653:   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)
5654: end;
5655:
5656:
5657: procedure SIRegister_uwinstr(CL: TPSCompiler);
5658: begin

```

```

5659: Function StrDec( S : String ) : String
5660: Function uIsNumeric( var S : String; var X : Float ) : Boolean
5661: Function ReadNumFromEdit( Edit : TEdit ) : Float
5662: Procedure WriteNumToFile( var F : Text; X : Float )
5663: end;
5664:
5665: procedure SIRegister_utexplot(CL: TPSPPascalCompiler);
5666: begin
5667:   Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean ) : Boolean
5668:   Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean )
5669:   Procedure TeX_LeaveGraphics( Footer : Boolean )
5670:   Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float )
5671:   Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float )
5672:   Procedure TeX_SetGraphTitle( Title : String )
5673:   Procedure TeX_SetOxTitle( Title : String )
5674:   Procedure TeX_SetOyTitle( Title : String )
5675:   Procedure TeX_PlotOxAxis
5676:   Procedure TeX_PlotOyAxis
5677:   Procedure TeX_PlotGrid( Grid : TGrid )
5678:   Procedure TeX_WriteGraphTitle
5679:   Function TeX_SetMaxCurv( NCurv : Byte ) : Boolean
5680:   Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer )
5681:   Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean )
5682:   Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String )
5683:   Procedure TeX_SetCurvStep( CurvIndex, Step : Integer )
5684:   Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer )
5685:   Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer )
5686:   Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer )
5687:   Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean )
5688:   Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix )
5689:   Function Xcm( X : Float ) : Float
5690:   Function Ycm( Y : Float ) : Float
5691: end;
5692:
5693: *-----*)
5694: procedure SIRegister_VarRecUtils(CL: TPSPPascalCompiler);
5695: begin
5696:   TConstArray', 'array of TVarRec
5697:   Function CopyVarRec( const Item : TVarRec ) : TVarRec
5698:   Function CreateConstArray( const Elements : array of const ) : TConstArray
5699:   Procedure FinalizeVarRec( var Item : TVarRec )
5700:   Procedure FinalizeConstArray( var Arr : TConstArray )
5701: end;
5702:
5703: procedure SIRegister_StStrS(CL: TPSPPascalCompiler);
5704: begin
5705:   Function HexBS( B : Byte ) : ShortString
5706:   Function HexWS( W : Word ) : ShortString
5707:   Function HexLS( L : LongInt ) : ShortString
5708:   Function HexPtrs( P : Pointer ) : ShortString
5709:   Function BinaryBS( B : Byte ) : ShortString
5710:   Function BinaryWS( W : Word ) : ShortString
5711:   Function BinaryLS( L : LongInt ) : ShortString
5712:   Function OctalBS( B : Byte ) : ShortString
5713:   Function OctalWS( W : Word ) : ShortString
5714:   Function OctalLS( L : LongInt ) : ShortString
5715:   Function Str2Int16S( const S : ShortString; var I : SmallInt ) : Boolean
5716:   Function Str2WordS( const S : ShortString; var I : Word ) : Boolean
5717:   Function Str2LongS( const S : ShortString; var I : LongInt ) : Boolean
5718:   Function Str2RealS( const S : ShortString; var R : Double ) : Boolean
5719:   Function Str2RealS( const S : ShortString; var R : Real ) : Boolean
5720:   Function Str2ExtS( const S : ShortString; var R : Extended ) : Boolean
5721:   Function Long2StrS( L : LongInt ) : ShortString
5722:   Function Real2StrS( R : Double; Width : Byte; Places : ShortInt ) : ShortString
5723:   Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt ) : ShortString
5724:   Function ValPrepS( const S : ShortString ) : ShortString
5725:   Function CharStrS( C : AnsiChar; Len : Cardinal ) : ShortString
5726:   Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5727:   Function PadS( const S : ShortString; Len : Cardinal ) : ShortString
5728:   Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5729:   Function LeftPadS( const S : ShortString; Len : Cardinal ) : ShortString
5730:   Function TrimLeadS( const S : ShortString ) : ShortString
5731:   Function TrimTrailsS( const S : ShortString ) : ShortString
5732:   Function Trims( const S : ShortString ) : ShortString
5733:   Function TrimSpacesS( const S : ShortString ) : ShortString
5734:   Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5735:   Function Centers( const S : ShortString; Len : Cardinal ) : ShortString
5736:   Function EntabS( const S : ShortString; TabSize : Byte ) : ShortString
5737:   Function DetabS( const S : ShortString; TabSize : Byte ) : ShortString
5738:   Function ScrambleS( const S, Key : ShortString ) : ShortString
5739:   Function SubstituteS( const S, FromStr,ToStr : ShortString ) : ShortString
5740:   Function Filters( const S, Filters : ShortString ) : ShortString
5741:   Function CharExistsS( const S : ShortString; C : AnsiChar ) : Boolean
5742:   Function CharCounts( const S : ShortString; C : AnsiChar ) : Byte
5743:   Function WordCounts( const S, WordDelims : ShortString ) : Cardinal
5744:   Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal ) : Boolean
5745:   Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString ) : ShortString
5746:   Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar ) : Cardinal
5747:   Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean

```

```

5748: Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar) : ShortString
5749: Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
Margin:Cardinal;PadToMargin:Boolean)
5750: Function CompStringS( const S1, S2 : ShortString) : Integer
5751: Function CompUCStringS( const S1, S2 : ShortString) : Integer
5752: Function SoundexS( const S : ShortString) : ShortString
5753: Function MakeLetterSetS( const S : ShortString) : Longint
5754: Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5755: Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5756: Function BMSearchUCS (var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
Pos:Cardinal):Bool;
5757: Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5758: Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5759: Function JustFilenameS( const PathName : ShortString) : ShortString
5760: Function JustNameS( const PathName : ShortString) : ShortString
5761: Function JustExtensionS( const Name : ShortString) : ShortString
5762: Function JustPathnameS( const PathName : ShortString) : ShortString
5763: Function AddBackSlashS( const DirName : ShortString) : ShortString
5764: Function CleanPathNameS( const PathName : ShortString) : ShortString
5765: Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5766: Function CommaizeS( L : LongInt) : ShortString
5767: Function CommaizeChS( L : LongInt; Ch : AnsiChar) : ShortString
5768: Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
DecPt:Char):ShortString;
5769: Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
RtCurr:ShortString;Sep:AnsiChar):ShortString;
5770: Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5771: Function StrrPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5772: Function StrrCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5773: Function StrrInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5774: Function StrrInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5775: Function StrrDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5776: Function StrrDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5777: Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5778: Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5779: Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5780: Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5781: Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5782: Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5783: Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
SubString:ShortString):Bool;
5784: Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
SubStr:ShortString):Bool;
5785: Function CopyFromToWordsS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5786: Function DeleteFromToWordsS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5787: Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5788: Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5789: Function ExtractTokensS(const S,
Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5790: Function IsChAlphaS( C : Char) : Boolean
5791: Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5792: Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5793: Function IsStrAlphaS( const S : Shortstring) : Boolean
5794: Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5795: Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5796: Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5797: Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5798: Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5799: Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5800: Function KeepCharsS( const S, Chars : ShortString) : ShortString
5801: Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
Cardinal):ShortString;
5802: Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
Replacements:Cardinal):ShortString;
5803: Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5804: Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
Replacements:Cardinal):ShortString
5805: Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
Replacements:Cardinal):ShortString
5806: Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5807: Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):bool
5808: Function TrimCharsS( const S, Chars : ShortString) : ShortString
5809: Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5810: end;
5811:
5812:
5813: *****unit uPSI_StUtils; from Systools4*****
5814: Function SignL( L : LongInt) : Integer
5815: Function SignF( F : Extended) : Integer
5816: Function MinWord( A, B : Word) : Word
5817: Function MidWord( W1, W2, W3 : Word) : Word
5818: Function MaxWord( A, B : Word) : Word
5819: Function MinLong( A, B : LongInt) : LongInt
5820: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5821: Function MaxLong( A, B : LongInt) : LongInt
5822: Function MinFloat( F1, F2 : Extended) : Extended

```

```

5823: Function MidFloat( F1, F2, F3 : Extended ) : Extended
5824: Function MaxFloat( F1, F2 : Extended ) : Extended
5825: Function MakeInteger16( H, L : Byte ) : SmallInt
5826: Function MakeWordsS( H, L : Byte ) : Word
5827: Function SwapNibble( B : Byte ) : Byte
5828: Function SwapWord( L : LongInt ) : LongInt
5829: Procedure SetFlag( var Flags : Word; FlagMask : Word )
5830: Procedure ClearFlag( var Flags : Word; FlagMask : Word )
5831: Function FlagIsSet( Flags, FlagMask : Word ) : Boolean
5832: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte )
5833: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte )
5834: Function ByteFlagIsSet( Flags, FlagMask : Byte ) : Boolean
5835: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt )
5836: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt )
5837: Function LongFlagIsSet( Flags, FlagMask : LongInt ) : Boolean
5838: Procedure ExchangeBytes( var I, J : Byte )
5839: Procedure ExchangeWords( var I, J : Word )
5840: Procedure ExchangeLongInts( var I, J : LongInt )
5841: Procedure ExchangeStructs( var I, J, Size : Cardinal )
5842: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word )
5843: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal )
5844: Function AddWordToPtr( P : __Pointer; W : Word ) : __Pointer
5845: //*****unit uPSI_StFIN;*****
5846: Function AccruedInterestMaturity( Issue, Maturity:TStDate; Rate, Par:Extended; Basis: TStBasis ) : Extended
5847: Function AccruedInterestPeriodic( Issue, Settlement, Maturity:TStDate; Rate,
Par:Extended; Frequency:TStFrequency; Basis : TStBasis ) : Extended
5848: Function BondDuration( Settlement, Maturity:TStDate; Rate,
Yield:Ext; Frequency:TStFrequency; Basis:TStBasis ) : Extended;
5849: Function BondPrice( Settlement, Maturity:TStDate; Rate, Yield, Redempt:Ext; Freq:TStFrequency; Basis:TStBasis ) : Extended
5850: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod : Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5851: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod : Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5852: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis ) : LongInt
5853: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer ) : Extended
5854: Function DiscountRate( Settlement, Maturity:TStDate; Price, Redemption:Extended; Basis:TStBasis ) : Extended;
5855: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer ) : Extended
5856: Function DollarToDecimalText( DecDollar : Extended ) : string
5857: Function DollarToFraction( DecDollar : Extended; Fraction : Integer ) : Extended
5858: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer ) : string
5859: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency ) : Extended
5860: Function FutureValueS( Rate:Extended; NPeriods:Int; Pmt,
PV:Extended; Freq:TStFreq; Timing:TStPaymentTime ) : Extended;
5861: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double ) : Extended
5862: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer ) : Extended
5863: Function InterestRateS( NPeriods:Int; Pmt, PV,
FV:Extended; Freq:TStFrequency; Timing:TStPaymentTime; Guess:Extended ) : Extended;
5864: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended ) : Extended
5865: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended ) : Extended
5866: Function IsCardValid( const S : string ) : Boolean
5867: Function ModifiedDuration( Settlement, Maturity:TStDate; Rate,
Yield:Extended; Freq:TStFrequency; Basis:TStBasis ) : Extended;
5868: Function ModifiedIRR( const Values: array of Double; FinanceRate, ReinvestRate: Extended ) : Extended
5869: Function ModifiedIRR16( const Values, NValues:Integer; FinanceRate, ReinvestRate: Extended ) : Extended
5870: Function NetPresentValueS( Rate : Extended; const Values : array of Double ) : Extended
5871: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer ) : Extended
5872: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency ) : Extended
5873: Function NonperiodicIRR( const Values:array of Double;const Dates:array of TStDate;Guess:Extended ) : Extended;
5874: Function NonperiodicNPV( Rate:Extended;const Values: array of Double;const Dates:array of TStDate ) : Extended;
5875: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
: TStPaymentTime ) : Extended
5876: Function Periods( Rate:Extended; Pmt, PV, FV:Extended; Frequency:TStFrequency; Timing:TStPaymentTime ) : Int;
5877: Function PresentValueS( Rate : Extended; NPeriods : Integer; Pmt, FV : Extended; Frequency : TStFrequency;
Timing : TStPaymentTime ) : Extended
5878: Function ReceivedAtMaturity( Settlement, Maturity:TStDate; Invest, Discount:Extended; Basis:TStBasis ) : Extended;
5879: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean ) : Extended
5880: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5881: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5882: Function TBillyYield( Settlement, Maturity : TStDate; Price : Extended ) : Extended
5883: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
Factor : Extended; NoSwitch : boolean ) : Extended
5884: Function YieldDiscounted( Settlement, Maturity:TStDate; Price, Redemption:Exted; Basis:TStBasis ) : Extended;
5885: Function YieldPeriodic( Settlement, Maturity:TStDate; Rate, Price,
Redemption:Extended; Freq:TStFrequency; Basis:TStBasis ) : Extended
5886: Function YieldMaturity( Issue, Settlement, Maturity:TStDate; Rate, Price:Extended; Basis:TStBasis ) : Extended;
5887: //*****unit uPSI_StAstroP;
5888: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray )
5889: //****unit unit uPSI_StStat; Statistic Package of SysTools*****
5890: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****
5891: Function AveDev( const Data : array of Double ) : Double
5892: Function AveDev16( const Data, NData : Integer ) : Double
5893: Function Confidence( Alpha, StandardDev : Double; Size : LongInt ) : Double
5894: Function Correlation( const Data1, Data2 : array of Double ) : Double
5895: Function Correlation16( const Data1, Data2, NData : Integer ) : Double
5896: Function Covariance( const Data1, Data2 : array of Double ) : Double
5897: Function Covariance16( const Data1, Data2, NData : Integer ) : Double
5898: Function DevSq( const Data : array of Double ) : Double
5899: Function DevSq16( const Data, NData : Integer ) : Double

```

```

5900: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5901: //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5902: Function GeometricMeanS( const Data : array of Double) : Double
5903: Function GeometricMean16( const Data, NData : Integer) : Double
5904: Function HarmonicMeanS( const Data : array of Double) : Double
5905: Function HarmonicMean16( const Data, NData : Integer) : Double
5906: Function Largest( const Data : array of Double; K : Integer) : Double
5907: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5908: Function MedianS( const Data : array of Double) : Double
5909: Function Median16( const Data, NData : Integer) : Double
5910: Function Mode( const Data : array of Double) : Double
5911: Function Mode16( const Data, NData : Integer) : Double
5912: Function Percentile( const Data : array of Double; K : Double) : Double
5913: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5914: Function PercentRank( const Data : array of Double; X : Double) : Double
5915: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5916: Function Permutations( Number, NumberChosen : Integer) : Extended
5917: Function Combinations( Number, NumberChosen : Integer) : Extended
5918: Function Factorials( N : Integer) : Extended
5919: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5920: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5921: Function Smallest( const Data : array of Double; K : Integer) : Double
5922: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5923: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5924: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5925: AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5926: +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSE: Double; F0 : Double; df : Integer;end
5927: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
LF:TStLinEst;ErrorStats:Bool;
5928: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
LF:TStLinEst;ErrorStats:Bool;
5929: Function Forecast(X: Double; const KnownY : array of Double; const KnownX : array of Double) : Double
5930: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double
5931: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5932: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5933: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5934: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5935: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5936: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5937: Function BinomDist( Numbers, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5938: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5939: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5940: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5941: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5942: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5943: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5944: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5945: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5946: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5947: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5948: Function NormSDist( Z : Single) : Single
5949: Function NormSInv( Probability : Single) : Single
5950: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5951: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5952: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5953: Function Erfc( X : Single) : Single
5954: Function GammaLn( X : Single) : Single
5955: Function LargestSort( const Data : array of Double; K : Integer) : Double
5956: Function SmallestSort( const Data : array of double; K : Integer) : Double
5957:
5958: procedure SIRegister_TStSorter(CL: TPSPPascalCompiler);
5959: Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5960: Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5961: Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5962: Function DefaultMergeName( MergeNum : Integer) : string
5963: Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5964:
5965: procedure SIRegister_StAstro(CL: TPSPPascalCompiler);
5966: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5967: Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5968: Function SunPos( UT : TStDateTimeRec) : TStPosRec
5969: Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5970: Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5971: Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5972: Function LunarPhase( UT : TStDateTimeRec) : Double
5973: Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5974: Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5975: Function FirstQuarter( D : TStDate) : TStLunarRecord
5976: Function FullMoon( D : TStDate) : TStLunarRecord
5977: Function LastQuarter( D : TStDate) : TStLunarRecord
5978: Function NewMoon( D : TStDate) : TStLunarRecord
5979: Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5980: Function NextFullMoon( D : TStDate) : TStDateTimeRec
5981: Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5982: Function NextNewMoon( D : TStDate) : TStDateTimeRec
5983: Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5984: Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5985: Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5986: Function PrevNewMoon( D : TStDate) : TStDateTimeRec

```

```

5987: Function SiderealTime( UT : TStDateTimeRec ) : Double
5988: Function Solstice( Y, Epoch : Integer; Summer : Boolean ) : TStDateTimeRec
5989: Function Equinox( Y, Epoch : Integer; Vernal : Boolean ) : TStDateTimeRec
5990: Function SEaster( Y, Epoch : Integer ) : TStDate
5991: Function DateToAJD( D : TDateTime ) : Double
5992: Function HoursMin( RA : Double ) : ShortString
5993: Function DegrMin( DC : Double ) : ShortString
5994: Function AJDToDate( D : Double ) : TDateTime
5995:
5996: Procedure SIRRegister_StDate(CL: TPSPascalCompiler);
5997: Function CurrentDate : TStDate
5998: Function StValidDate( Day, Month, Year, Epoch : Integer ) : Boolean
5999: Function DMYtoStDate( Day, Month, Year, Epoch : Integer ) : TStDate
6000: Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer )
6001: Function StIncDate( Julian : TStDate; Days, Months, Years : Integer ) : TStDate
6002: Function IncDateTrunc( Julian : TStDate; Months, Years : Integer ) : TStDate
6003: Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer )
6004: Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType ) : TStDate
6005: Function WeekOfYear( Julian : TStDate ) : Byte
6006: Function AstJulianDate( Julian : TStDate ) : Double
6007: Function AstJulianDateToStDate( AstJulian : Double; Truncate : Boolean ) : TStDate
6008: Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime ) : Double
6009: Function StDayOfWeek( Julian : TStDate ) : TStDayType
6010: Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer ) : TStDayType
6011: Function StIsLeapYear( Year : Integer ) : Boolean
6012: Function StDaysInMonth( Month : Integer; Year, Epoch : Integer ) : Integer
6013: Function ResolveEpoch( Year, Epoch : Integer ) : Integer
6014: Function ValidTime( Hours, Minutes, Seconds : Integer ) : Boolean
6015: Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte )
6016: Function HMSToStTime( Hours, Minutes, Seconds : Byte ) : TStTime
6017: Function CurrentTime : TStTime
6018: Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte )
6019: Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte ) : TStTime
6020: Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte ) : TStTime
6021: Function RoundToNearestHour( T : TStTime; Truncate : Boolean ) : TStTime
6022: Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean ) : TStTime
6023: Procedure DateTimeDiff( const DT1:TStDateTimeRec; var DT2:TStDateTimeRec; var Days:LongInt; var Secs:LongInt )
6024: Procedure IncDateTime( const DT1:TStDateTimeRec; var DT2:TStDateTimeRec; Days:Integer; Secs:LongInt )
6025: Function DateTimeToStDate( DT : TDateTime ) : TStDate
6026: Function DateTimeToStTime( DT : TDateTime ) : TStTime
6027: Function StDateToDate( D : TStDate ) : TDateTime
6028: Function StTimeToDate( T : TStTime ) : TDateTime
6029: Function Convert2ByteDate( TwoByteDate : Word ) : TStDate
6030: Function Convert4ByteDate( FourByteDate : TStDate ) : Word
6031:
6032: Procedure SIRRegister_StDateSt(CL: TPSPascalCompiler);
6033: Function DateStringHMSToAstJD( const Picture, DS : string; H, M, S, Epoch : integer ) : Double
6034: Function MonthToString( const Month : Integer ) : string
6035: Function DateStringToStDate( const Picture, S : string; Epoch : Integer ) : TStDate
6036: Function DateStringToDMY( const Picture, S:string; Epoch:Integer; var D, M, Y : Integer ):Boolean
6037: Function StDateToString( const Picture : string; const Julian : TStDate; Pack : Boolean ):string
6038: Function DayOfWeekToString( const WeekDay : TStDayType ) : string
6039: Function DMYToDateString( const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string
6040: Function CurrentDateString( const Picture : string; Pack : Boolean ) : string
6041: Function CurrentTimeString( const Picture : string; Pack : Boolean ) : string
6042: Function TimeStringToHMS( const Picture, S : string; var H, M, S : Integer ) : Boolean
6043: Function TimeStringToStTime( const Picture, S : string ) : TStTime
6044: Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
6045: Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
6046: Function DateStringIsBlank( const Picture, S : string ) : Boolean
6047: Function InternationalDate( ForceCentury : Boolean ) : string
6048: Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean ) : string
6049: Function InternationalTime( ShowSeconds : Boolean ) : string
6050: Procedure ResetInternationalInfo
6051:
6052: procedure SIRRegister_StBase(CL: TPSPascalCompiler);
6053: Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer ) : Boolean
6054: Function AnsiUpperCaseShort32( const S : string ) : string
6055: Function AnsiCompareTextShort32( const S1, S2 : string ) : Integer
6056: Function AnsiCompareStrShort32( const S1, S2 : string ) : Integer
6057: Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer ) : Longint
6058: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer, OutLen:LongInt ) : Longint
6059: Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte )
6060: Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal )
6061: Function Upcase( C : AnsiChar ) : AnsiChar
6062: Function LoCase( C : AnsiChar ) : AnsiChar
6063: Function CompareLetterSets( Set1, Set2 : Longint ) : Cardinal
6064: Function CompStruct( const S1, S2, Size : Cardinal ) : Integer
6065: Function Search( const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6066: Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Bool
6067: Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Bool
6068: Function IsOrInheritsFrom( Root, Candidate : TClass ) : boolean
6069: Procedure RaiseContainerError( Code : longint )
6070: Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const )
6071: Function ProductOverflow( A, B : Longint ) : Boolean
6072: Function StNewStr( S : string ) : PShortString
6073: Procedure StDisposeStr( PS : PShortString )
6074: Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer )
6075: Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer )

```

```

6076: Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6077: Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6078: Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6079: Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6080:
6081: procedure SIRegister_usvd(CL: TPSPPascalCompiler);
6082: begin
6083: Procedure SV_DecomP( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6084: Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6085: Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6086: Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6087: Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6088: end;
6089:
6090: //*****unit unit ; StMath Package of SysTools*****
6091: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6092: Function PowerS( Base, Exponent : Extended) : Extended
6093: Function StInvCos( X : Double) : Double
6094: Function StInvSin( Y : Double) : Double
6095: Function StInvTan2( X, Y : Double) : Double
6096: Function StTan( A : Double) : Double
6097: Procedure DumpException; //unit StExpEng;
6098: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6099:
6100: //*****unit unit ; StCRC Package of SysTools*****
6101: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6102: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6103: Function Adler32OfFile( FileName : AnsiString) : LongInt
6104: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6105: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6106: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6107: Function Crc32Prim( var Data, DataSize, CurCrc : LongInt) : LongInt
6108: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6109: Function Crc32OfFile( FileName : AnsiString) : LongInt
6110: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6111: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6112: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6113: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6114: Function Kermit16OfFile( Stream : TStream; CurCrc : Cardinal) : Cardinal
6115: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6116:
6117: //*****unit unit ; StBCD Package of SysTools*****
6118: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6119: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6120: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6121: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6122: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6123: Function NegBcd( const B : TbcdS) : TbcdS
6124: Function AbsBcd( const B : TbcdS) : TbcdS
6125: Function FracBcd( const B : TbcdS) : TbcdS
6126: Function IntBcd( const B : TbcdS) : TbcdS
6127: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6128: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6129: Function ValBcd( const S : string) : TbcdS
6130: Function LongBcd( L : LongInt) : TbcdS
6131: Function ExtBcd( E : Extended) : TbcdS
6132: Function ExpBcd( const B : TbcdS) : TbcdS
6133: Function LnBcd( const B : TbcdS) : TbcdS
6134: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6135: Function PowBcd( const B, E : TbcdS) : TbcdS
6136: Function SqrtBcd( const B : TbcdS) : TbcdS
6137: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6138: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6139: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6140: Function IsIntBcd( const B : TbcdS) : Boolean
6141: Function TruncBcd( const B : TbcdS) : LongInt
6142: Function BcdExt( const B : TbcdS) : Extended
6143: Function RoundBcd( const B : TbcdS) : LongInt
6144: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6145: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6146: Function FormatBcd( const Format : string; const B : TbcdS) : string
6147: Function StrGeneralBcd( const B : TbcdS) : string
6148: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6149: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6150:
6151: //*****unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*****
6152: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6153: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6154: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6155: Function StDeEscape( const EscStr : AnsiString) : Char
6156: Function StDoEscape( Delim : Char) : AnsiString
6157: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6158: Function AnsiHashText( const S : string; Size : Integer) : Integer
6159: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6160: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6161: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6162:
6163: //*****unit unit ; StNetCon Package of SysTools*****
6164: with AddClassN(FindClass('TStComponent')),'TStNetConnection') do begin

```

```

6165: Constructor Create( AOwner : TComponent )
6166: Function Connect : DWord
6167: Function Disconnect : DWord
6168: RegisterProperty('Password', 'String', iptrw);
6169: Property('UserName', 'String', iptrw);
6170: Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6171: Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6172: Property('LocalDevice', 'String', iptrw);
6173: Property('ServerName', 'String', iptrw);
6174: Property('ShareName', 'String', iptrw);
6175: Property('OnConnect', 'TNotifyEvent', iptrw);
6176: Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6177: Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6178: Property('OnDisconnect', 'TNotifyEvent', iptrw);
6179: Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6180: Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6181: end;
6182: //***** Thread Functions Context of Win API --- more objects in SyncObjs.pas
6183: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6184: Procedure InitializeCriticalSection( var lpCriticalSection : TRTCriticalSection)
6185: Procedure EnterCriticalSection( var lpCriticalSection : TRTCriticalSection)
6186: Procedure LeaveCriticalSection( var lpCriticalSection : TRTCriticalSection)
6187: Function InitializeCriticalSectionAndSpinCount(var
lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):BOOL;
6188: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):DWORD;
6189: Function TryEnterCriticalSection( var lpCriticalSection : TRTCriticalSection) : BOOL
6190: Procedure DeleteCriticalSection( var lpCriticalSection : TRTCriticalSection)
6191: Function GetThreadContext( hThread : THandle; var lpContext : TContext ) : BOOL
6192: Function SetThreadContext( hThread : THandle; const lpContext : TContext ) : BOOL
6193: Function SuspendThread( hThread : THandle ) : DWORD
6194: Function ResumeThread( hThread : THandle ) : DWORD
6195: Function CreateThread2( ThreadFunc: TThreadFunction2; thrid: DWord ) : THandle
6196: Function GetCurrentThread : THandle
6197: Procedure ExitThread( dwExitCode : DWORD )
6198: Function TerminateThread( hThread : THandle; dwExitCode : DWORD ) : BOOL
6199: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD ) : BOOL
6200: Procedure EndThread(ExitCode: Integer);
6201: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD ) : DWORD
6202: Function MakeProcInstance( Proc : FARPROC; Instance : THandle ) : FARPROC
6203: Procedure FreeProcInstance( Proc : FARPROC )
6204: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD )
6205: Function DisableThreadLibraryCalls( hLibModule : HMODULE ) : BOOL
6206: Procedure ParallelJob1( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean );
6207: Procedure ParallelJob1( ATarg : Pointer; AParam : Pointer; ASafeSection : boolean );
6208: Procedure ParallelJob2( AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool );
6209: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean );
6210: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6211: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6212: Function CurrentParallelJobInfo : TParallelJobInfo
6213: Function ObtainParallelJobInfo : TParallelJobInfo
6214: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo );
6215: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD ) : BOOL';
6216: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle ) : BOOL';
6217: Function
DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL';
6218: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL';
6219: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD ) : BOOL';
6220: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD ) : BOOL';
6221: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD ) : BOOL';
6222:
6223: *****unit uPSI_JclMime;
6224: Function MimeEncodeString( const S : AnsiString ) : AnsiString
6225: Function MimeDecodeString( const S : AnsiString ) : AnsiString
6226: Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6227: Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6228: Function MimeEncodedSize( const I : Cardinal ) : Cardinal
6229: Function MimeDecodedSize( const I : Cardinal ) : Cardinal
6230: Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6231: Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6232: Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6233: Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card) :
Cardinal;
6234:
6235: *****unit uPSI_JclPrint;
6236: Procedure DirectPrint( const Printer, Data : string)
6237: Procedure SetPrinterPixelsPerInch
6238: Function GetPrinterResolution : TPoint
6239: Function CharFitsWithinDots( const Text : string; const Dots : Integer ) : Integer
6240: Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6241:
6242:
6243: *****unit uPSI_ShLwApi;*****
6244: Function StrChr( lpStart : PChar; wMatch : WORD ) : PChar
6245: Function StrChri( lpStart : PChar; wMatch : WORD ) : PChar
6246: Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer ) : Integer
6247: Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer ) : Integer

```

```

6248: Function StrCSpn( lpStr_ , lpSet : PChar ) : Integer
6249: Function StrCSpnI( lpStr1, lpSet : PChar ) : Integer
6250: Function StrDup( lpSrch : PChar ) : PChar
6251: Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT ) : PChar
6252: Function StrFormatKBSIZE( qdw : Dword; szBuf : PChar; uiBufSize : UINT ) : PChar
6253: Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer) : Integer
6254: Function StrIsIntlEqual(fCaseSens: BOOL; lpString1, lpString2: PChar; nChar : Integer) : BOOL
6255: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6256: Function StrPBrk( psz, pszSet : PChar) : PChar
6257: Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD ) : PChar
6258: Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD ) : PChar
6259: Function StrRstrI( lpSource, lpLast, lpSrch : PChar) : PChar
6260: Function StrSpn( psz, pszSet : PChar) : Integer
6261: Function StrStr( lpFirst, lpSrch : PChar) : PChar
6262: Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6263: Function StrToInt( lpSrch : PChar) : Integer
6264: Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6265: Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6266: Function ChrCmpI( w1, w2 : WORD) : BOOL
6267: Function ChrCmpIA( w1, w2 : WORD) : BOOL
6268: Function ChrCmpIW( w1, w2 : WORD) : BOOL
6269: Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6270: Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6271: Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBufferSize : Integer) : PChar
6272: Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6273: Function IntlStrEqWorker(fCaseSens: BOOL; lpString1, lpString2:PChar; nChar: Integer) : BOOL
6274: Function IntlstrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6275: SZ_CONTENTTYPE_HTML,'String 'text/html'
6276: SZ_CONTENTTYPE_HTMLW,'String 'text/html'
6277: SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA';
6278: SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6279: SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6280: SZ_CONTENTTYPE_CDF','String SZ_CONTENTTYPE_CDFA';
6281: Function PathIsHTMLFile( pszPath : PChar) : BOOL
6282: STIF_DEFAULT','LongWord( $00000000);
6283: STIF_SUPPORT_HEX','LongWord( $00000001);
6284: Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6285: Function StrNCPy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6286: Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6287: Function PathAddBackslash( pszPath : PChar) : PChar
6288: Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6289: Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6290: Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6291: Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6292: Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6293: Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6294: Function PathCompactPathEx(pszOut: PChar; pszSrc: PChar; cchMax: UINT; dwFlags:DWORD) : BOOL
6295: Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6296: Function PathFileExists( pszPath : PChar) : BOOL
6297: Function PathFindExtension( pszPath : PChar) : PChar
6298: Function PathFindFileName( pszPath : PChar) : PChar
6299: Function PathFindNextComponent( pszPath : PChar) : PChar
6300: Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6301: Function PathGetArgs( pszPath : PChar) : PChar
6302: Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer) : PChar
6303: Function PathIsLFNfileSpec( lpName : PChar) : BOOL
6304: Function PathGetCharType( ch : Char) : UINT
6305: GCT_INVALID','LongWord( $0000);
6306: GCT_LFNCHAR','LongWord( $0001);
6307: GCT_SHORTCHAR','LongWord( $0002);
6308: GCT_WILD','LongWord( $0004);
6309: GCT_SEPARATOR','LongWord( $0008);
6310: Function PathGetDriveNumber( pszPath : PChar) : Integer
6311: Function PathIsDirectory( pszPath : PChar) : BOOL
6312: Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6313: Function PathIsFileSpec( pszPath : PChar) : BOOL
6314: Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6315: Function PathIsRelative( pszPath : PChar) : BOOL
6316: Function PathIsRoot( pszPath : PChar) : BOOL
6317: Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6318: Function PathIsUNC( pszPath : PChar) : BOOL
6319: Function PathIsNetworkPath( pszPath : PChar) : BOOL
6320: Function PathIsUNCServer( pszPath : PChar) : BOOL
6321: Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6322: Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6323: Function PathIsURL( pszPath : PChar) : BOOL
6324: Function PathMakePretty( pszPath : PChar) : BOOL
6325: Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6326: Function PathParseIconLocation( pszIconFile : PChar) : Integer
6327: Procedure PathQuoteSpaces( lpsz : PChar)
6328: Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL
6329: Procedure PathRemoveArgs( pszPath : PChar)
6330: Function PathRemoveBackslash( pszPath : PChar) : PChar
6331: Procedure PathRemoveBlanks( pszPath : PChar)
6332: Procedure PathRemoveExtension( pszPath : PChar)
6333: Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6334: Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6335: Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6336: Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)

```

```

6337: Function PathSkipRoot( pszPath : PChar ) : PChar
6338: Procedure PathStripPath( pszPath : PChar )
6339: Function PathStripToRoot( pszPath : PChar ) : BOOL
6340: Procedure PathUnquoteSpaces( lpsz : PChar )
6341: Function PathMakeSystemFolder( pszPath : PChar ) : BOOL
6342: Function PathUnmakeSystemFolder( pszPath : PChar ) : BOOL
6343: Function PathIsSystemFolder( pszPath : PChar; dwAttrib : DWORD ) : BOOL
6344: Procedure PathUndecorate( pszPath : PChar )
6345: Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6346: URL_SCHEME_INVALID', 'LongInt'( - 1);
6347: URL_SCHEME_UNKNOWN', 'LongInt'( 0 );
6348: URL_SCHEME_FTP', 'LongInt'( 1 );
6349: URL_SCHEME_HTTP', 'LongInt'( 2 );
6350: URL_SCHEME_GOPHER', 'LongInt'( 3 );
6351: URL_SCHEME_MAILTO', 'LongInt'( 4 );
6352: URL_SCHEME_NEWS', 'LongInt'( 5 );
6353: URL_SCHEME_NNTP', 'LongInt'( 6 );
6354: URL_SCHEME_TELNET', 'LongInt'( 7 );
6355: URL_SCHEME_WAIS', 'LongInt'( 8 );
6356: URL_SCHEME_FILE', 'LongInt'( 9 );
6357: URL_SCHEME_MK', 'LongInt'( 10 );
6358: URL_SCHEME_HTTPS', 'LongInt'( 11 );
6359: URL_SCHEME_SHELL', 'LongInt'( 12 );
6360: URL_SCHEME_SNEWS', 'LongInt'( 13 );
6361: URL_SCHEME_LOCAL', 'LongInt'( 14 );
6362: URL_SCHEME_JAVASCRIPT', 'LongInt'( 15 );
6363: URL_SCHEME_VBSCRIPT', 'LongInt'( 16 );
6364: URL_SCHEME_ABOUT', 'LongInt'( 17 );
6365: URL_SCHEME_RES', 'LongInt'( 18 );
6366: URL_SCHEME_MAXVALUE', 'LongInt'( 19 );
6367: URL_SCHEME', 'Integer
6368: URL_PART_NONE', 'LongInt'( 0 );
6369: URL_PART_SCHEME', 'LongInt'( 1 );
6370: URL_PART_HOSTNAME', 'LongInt'( 2 );
6371: URL_PART_USERNAME', 'LongInt'( 3 );
6372: URL_PART_PASSWORD', 'LongInt'( 4 );
6373: URL_PART_PORT', 'LongInt'( 5 );
6374: URL_PART_QUERY', 'LongInt'( 6 );
6375: URL_PART', 'DWORD
6376: URLIs_URL', 'LongInt'( 0 );
6377: URLIs_OPAQUE', 'LongInt'( 1 );
6378: URLIs_NOHISTORY', 'LongInt'( 2 );
6379: URLIs_FILEURL', 'LongInt'( 3 );
6380: URLIs_APPLICABLE', 'LongInt'( 4 );
6381: URLIs_DIRECTORY', 'LongInt'( 5 );
6382: URLIs_HASQUERY', 'LongInt'( 6 );
6383: TURLIs', 'DWORD
6384: URL_UNESCAPE', 'LongWord( $10000000 );
6385: URL_ESCAPE_UNSAFE', 'LongWord( $20000000 );
6386: URL_PLUGGABLE_PROTOCOL', 'LongWord( $40000000 );
6387: URL_WININET_COMPATIBILITY', 'LongWord( DWORD( $80000000 ) );
6388: URL_DONT_ESCAPE_EXTRA_INFO', 'LongWord( $02000000 );
6389: URL_ESCAPE_SPACES_ONLY', 'LongWord( $04000000 );
6390: URL_DONT_SIMPLIFYT', 'LongWord( $08000000 );
6391: URL_NO_META', 'longword( URL_DONT_SIMPLIFY );
6392: URL_UNESCAPE_INPLACE', 'LongWord( $00100000 );
6393: URL_CONVERT_IF_DOSPATH', 'LongWord( $00200000 );
6394: URL_UNESCAPE_HIGH_ANSI_ONLY', 'LongWord( $00400000 );
6395: URL_INTERNAL_PATH', 'LongWord( $00800000 );
6396: URL_FILE_USE_PATHURL', 'LongWord( $00001000 );
6397: URL_ESCAPE_PERCENT', 'LongWord( $00001000 );
6398: URL_ESCAPE_SEGMENT_ONLY', 'LongWord( $00002000 );
6399: URL_PARTFLAG_KEEPSCHEME', 'LongWord( $00000001 );
6400: URL_APPLY_DEFAULT', 'LongWord( $00000001 );
6401: URL_APPLY_GUESSSCHEME', 'LongWord( $00000002 );
6402: URL_APPLY_GUESSFILE', 'LongWord( $00000004 );
6403: URL_APPLY_FORCEAPPLY', 'LongWord( $00000008 );
6404: Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL ) : Integer
6405: Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD) : HRESULT;
6406: Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD) : HRESULT;
6407: Function UrlIsOpaque( pszURL : PChar ) : BOOL
6408: Function UrlIsNoHistory( pszURL : PChar ) : BOOL
6409: Function UrlIsFileUrl( pszURL : PChar ) : BOOL
6410: Function UrlIs( pszUrl : PChar; URLIs : TURLIs ) : BOOL
6411: Function UrlGetLocation( psz1 : PChar ) : PChar
6412: Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD ) : HRESULT
6413: Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6414: Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6415: Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6416: Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6417: Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6418: Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6419: Function HashData( pData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6420: Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD ) : HRESULT
6421: Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD ) : HRESULT
6422: Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6423: Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6424: Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar ) : DWORD
6425: Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD ) : Longint

```

```

6426: Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
6427:   pcchValueName:DWORD;pdwType:DWORD; pvData : Pointer; pcbData : DWORD ) : Longint;
6428: Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcch.MaxValueNameLen:DWORD):Longint;
6429: Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6429: Function SHRegGetPath( hKey:HKEY; ppszSubKey,ppszValue: PChar; pszPath: PChar; dwFlags: DWORD) : DWORD
6430: Function SHRegSetPath( hKey:HKEY; ppszSubKey, ppszValue, ppszPath : PChar; dwFlags : DWORD) : DWORD
6431: SHREGDEL_DEFAULT', 'LongWord( $00000000);
6432: SHREGDEL_HKCU', 'LongWord( $00000001);
6433: SHREGDEL_HKLM', 'LongWord( $00000010);
6434: SHREGDEL_BOTH', 'LongWord( $00000011);
6435: SHREGENUM_DEFAULT', 'LongWord( $00000000);
6436: SHREGENUM_HKCU', 'LongWord( $00000001);
6437: SHREGENUM_HKLM', 'LongWord( $00000010);
6438: SHREGENUM_BOTH', 'LongWord( $00000011);
6439: SHREGSET_HKCU', 'LongWord( $00000001);
6440: SHREGSET_FORCE_HKCU', 'LongWord( $00000002);
6441: SHREGSET_HKLM', 'LongWord( $00000004);
6442: SHREGSET_FORCE_HKLM', 'LongWord( $00000008);
6443: TSHRegDelFlags', 'DWORD
6444: TSHRegEnumFlags', 'DWORD
6445: HUSKEY', 'THandle
6446: ASSOCF_INIT_NOREMAPCLSID', 'LongWord( $00000001);
6447: ASSOCF_INIT_BYEXENAME', 'LongWord( $00000002);
6448: ASSOCF_OPEN_BYEXENAME', 'LongWord( $00000002);
6449: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord( $00000004);
6450: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord( $00000008);
6451: ASSOCF_NOUSERSETTINGS', 'LongWord( $00000010);
6452: ASSOCF_NOTRUNCATE', 'LongWord( $00000020);
6453: ASSOCF_VERIFY', 'LongWord( $00000040);
6454: ASSOCF_REMAPRUNDLL', 'LongWord( $00000080);
6455: ASSOCF_NOFIXUPS', 'LongWord( $00000100);
6456: ASSOCF_IGNOREBASECLASS', 'LongWord( $00000200);
6457: ASSOCFT', 'DWORD
6458: ASSOCSTR_COMMAND', 'LongInt'( 1);
6459: ASSOCSTR_EXECUTABLE', 'LongInt'( 2);
6460: ASSOCSTR_FRIENDLYDOCNAME', 'LongInt'( 3);
6461: ASSOCSTR_FRIENDLYAPPNAME', 'LongInt'( 4);
6462: ASSOCSTR_NOOPEN', 'LongInt'( 5);
6463: ASSOCSTR_SHELLNEWVALUE', 'LongInt'( 6);
6464: ASSOCSTR_DDECOMMAND', 'LongInt'( 7);
6465: ASSOCSTR_DDEIFEXEC', 'LongInt'( 8);
6466: ASSOCSTR_DDEAPPLICATION', 'LongInt'( 9);
6467: ASSOCSTR_DDETOPIC', 'LongInt'( 10);
6468: ASSOCSTR_INFOTIP', 'LongInt'( 11);
6469: ASSOCSTR_MAX', 'LongInt'( 12);
6470: ASSOCSTR', 'DWORD
6471: ASSOCKEY_SHELLEXECCLASS', 'LongInt'( 1);
6472: ASSOCKEY_APP', 'LongInt'( 2);
6473: ASSOCKEY_CLASS', 'LongInt'( 3);
6474: ASSOCKEY_BASECLASS', 'LongInt'( 4);
6475: ASSOCKEY_MAX', 'LongInt'( 5);
6476: ASSOCKEY', 'DWORD
6477: ASSOCDATA_MSIDEScriptor', 'LongInt'( 1);
6478: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt'( 2);
6479: ASSOCDATA_QUERYCLASSTOOL', 'LongInt'( 3);
6480: ASSOCDATA_HASPERUSERASSOC', 'LongInt'( 4);
6481: ASSOCDATA_MAX', 'LongInt'( 5);
6482: ASSOCDATA', 'DWORD
6483: ASSOCENUM_NONE', 'LongInt'( 0);
6484: ASSOCENUM', 'DWORD
6485: SID_IQueryAssociations', 'String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6486: SHACF_DEFAULT,$00000000);
6487: SHACF_FILESYSTEM', 'LongWord( $00000001);
6488: SHACF_URLHISTORY', 'LongWord( $00000002);
6489: SHACF_URLMRU', 'LongWord( $00000004);
6490: SHACF_USETAB', 'LongWord( $00000008);
6491: SHACF_FILESYS_ONLY', 'LongWord( $00000010);
6492: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord( $10000000);
6493: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord( $20000000);
6494: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord( $40000000);
6495: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord( DWORD ( $80000000 ) );
6496: Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6497: Procedure SHSetThreadRef( punk : IUnknown)
6498: Procedure SHGetThreadRef( out ppunk : IUnknown)
6499: CTF_INSIST', 'LongWord( $00000001);
6500: CTF_THREAD_REF', 'LongWord( $00000002);
6501: CTF_PROCESS_REF', 'LongWord( $00000004);
6502: CTF_COINIT', 'LongWord( $00000008);
6503: Function SHCreateShellPalette( hdc : HDC ) : HPALETTE
6504: Procedure ColorRGBtoHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6505: Function ColorHLStoRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6506: Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6507: Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6508: Function DrawFocusRect( hdc : HDC; const lprc : TRect) : BOOL
6509: Function FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6510: Function FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6511: Function InvertRect( hdc : HDC; const lprc : TRect) : BOOL
6512: Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6513: Function SetRectEmpty( var lprc : TRect) : BOOL

```

```

6514: Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6515: Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6516: Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6517: Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6518:
6519: Function InitializeFlatSB( hWnd : HWND) : Bool
6520: Procedure UninitializeFlatSB( hWnd : HWND)
6521: Function FlatSB_SetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6522: Function GET_APPCOMMAND LPARAM( lParam : Integer) : Word //of JvWin32
6523: Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6524: Function GET_MOUSEKEY LPARAM( lParam : Integer) : Integer
6525: Function GET_FLAGS LPARAM( lParam : Integer) : Word
6526: Function GET_KEYSTATE LPARAM( lParam : Integer) : Word
6527: Function GET_PSI LPARAM( lParam : Integer) : Word
6528:
6529:
6530: // **** 204 unit uPSI_ShellAPI;
6531: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6532: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6533: Procedure DragFinish( Drop : HDROP)
6534: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6535: Function ShellExecute( hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6536: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6537: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6538: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6539: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6540: Function ExtractionIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6541: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6542: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6543: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6544: Procedure SHFreeNameMappings( hNameMappings : THandle)
6545:
6546: DLLVER_PLATFORM_WINDOWS,'LongWord( $00000001 );
6547: DLLVER_PLATFORM_NT,'LongWord( $00000002 );
6548: DLLVER_MAJOR_MASK','LongWord( Int64( $FFFF000000000000 ) );
6549: DLLVER_MINOR_MASK','LongWord( Int64( $0000FFF000000000 ) );
6550: DLLVER_BUILD_MASK','LongWord( Int64( $000000000FFF0000 ) );
6551: DLLVER_QFE_MASK','LongWord( Int64( $000000000000FFFF ) );
6552: Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6553: Function SimpleXMLEncode( const S : string) : string
6554: Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6555: Function XMLEncode( const S : string) : string
6556: Function XMLDecode( const S : string) : string
6557: Function EntityEncode( const S : string) : string
6558: Function EntityDecode( const S : string) : string
6559:
6560: procedure RIRegister_CPort_Routines(S: TPSEExec);
6561: Procedure EnumComPorts( Ports : TStrings)
6562: Procedure ListComPorts( Ports : TStrings)
6563: Procedure ComPorts( Ports : TStrings) //Alias to Arduino
6564: Function GetComPorts: TStringlist;
6565: Function StrToBaudRate( Str : string) : TBaudRate
6566: Function StrToStopBits( Str : string) : TStopBits
6567: Function StrToDataBits( Str : string) : TDataBits
6568: Function StrToParity( Str : string) : TParityBits
6569: Function StrToFlowControl( Str : string) : TFlowControl
6570: Function BaudRateToStr( BaudRate : TBaudRate) : string
6571: Function StopBitsToStr( StopBits : TStopBits) : string
6572: Function DataBitsToStr( DataBits : TDataBits) : string
6573: Function ParityToStr( Parity : TParityBits) : string
6574: Function FlowControlToStr( FlowControl : TFlowControl) : string
6575: Function ComErrorsToStr( Errors : TComErrors) : String
6576:
6577: Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6578: Function DispatchMessage( const lpMsg : TMsg) : Longint
6579: Function TranslateMessage( const lpMsg : TMsg) : BOOL
6580: Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6581: Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6582: Function GetMessagePos : DWORD
6583: Function GetMessageTime : Longint
6584: Function GetMessageExtraInfo : Longint
6585: Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6586: Procedure JAddToRecentDocs( const Filename : string)
6587: Procedure ClearRecentDocs
6588: Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6589: Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6590: Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6591: Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6592: Function RecycleFile( FileMode : string) : Boolean
6593: Function JCopyFile( FromFile, ToDir : string) : Boolean
6594: Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6595: Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6596: Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL
6597: Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL
6598: Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL
6599: Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD; dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD;pszGroupName: LP

```

```

6600: Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
6601:   dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
6602:   lpResumeHandle:DWORD; pszGroupName
6603:   Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
6604:   dwServiceState : DWORD;lpServices : LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
6605:   lpResumeHandle:DWORD; pszGroupName
6606:   Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6607:   **** unit uPSI_JclPeImage;
6608:   Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6609:   Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6610:   Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
6611:   DWORD) : TJclRebaseImageInfo
6612:   Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6613:   Function PeClearCheckSum( const FileName : TFileName) : Boolean
6614:   Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6615:   Function PeDoesExportFunction(const FileName:TFileName;const
6616:   FuncName:string;Options:TJclSmartCompOptions):Bool;
6617:   Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
6618:   ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6619:   Function PeIsExportFunctionForwarded(const FileName:TFileName;const
6620:   FunctionName:string;Options:TJclSmartCompOptions):Bool
6621:   Function PeDoesImportFunction ( const FileName : TFileName; const FunctionName : string; const LibraryName
6622:   : string; Options : TJclSmartCompOptions) : Boolean
6623:   Function PeDoesImportLibrary(const FileName:TFileName;const
6624:   LibraryName:string;Recursive:Boolean):Boolean);
6625:   Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
6626:   Boolean; FullName : Boolean) : Boolean
6627:   Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
6628:   LibraryName:string; IncludeLibNames : Boolean): Boolean
6629:   Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6630:   Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6631:   Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6632:   Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
6633:   NamesList:TStrings):Bool
6634:   Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6635:   Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullName,
6636:   Descr:Bool):Bool;
6637:   Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6638:   Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6639:   //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6640:   //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6641:   //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6642:   //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
6643:   PImageSectionHeader
6644:   //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6645:   //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6646:   Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
6647:   _Pointer;
6648:   SJRegister_TJclPeSectionStream(CL);
6649:   SJRegister_TJclPeMapImgHookItem(CL);
6650:   SJRegister_TJclPeMapImgHooks(CL);
6651:   //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
6652:   NtHeaders:TImageNtHeaders):Boolean
6653:   //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6654:   Type TJclBorUmSymbolKind', '(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6655:   TJclBorUmSymbolModifier', '(smQualified, smLinkProc )
6656:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6657:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6658:   TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6659:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6660:   Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
6661:   TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6662:   Function PeBorUmangleName1(const Name:string;var Unmangled:string;var
6663:   Description:TJclBorUmDescription):TJclBorUmResult;
6664:   Function PeBorUmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6665:   Function PeBorUmangleName3( const Name : string) : string;
6666:   Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6667:   Function PeUmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6668:   ****
6669:   Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6670:   Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6671:   Procedure DeleteVolumeLabel( Drive : Char) : Cardinal
6672:   //Procedure EnumerateDirectories(const
6673:   StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6674:   //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
6675:   IncludeItem:TIncludeItemFunc);
6676:   Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6677:   Function FileMatchesMask( const FileName, FileMode : AnsiString) : Boolean
6678:   Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6679:   Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6680:   Function FlushOsBuffers( Handle : Integer) : Boolean
6681:   Function GetCurrentUser : AnsiString
6682:   Function GetDiskClass( Drive : Char) : DiskClass

```

```

6668: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,  
SectorsPerCluster:Cardinal):Bool;
6669: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var  
DiskSize:Double):Bool;
6670: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var  
DiskSize:Comp):Boolean;
6671:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6672: Function getDiskSpace2(const path: String; index: integer): int64;
6673: Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6674: Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6675: Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6676: Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6677: Function GetLongPath( const APath : AnsiString) : AnsiString
6678: Function GetMachineName : AnsiString
6679: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6680: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6681: Function GetShortPath( const APath : AnsiString) : AnsiString
6682: Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6683: Function GetTempFolder( aForceSlash : boolean) : AnsiString
6684: Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6685: Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6686: Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6687: Function StIsDirectory( const DirName : AnsiString) : Boolean
6688: Function IsDirectoryEmpty( const S : AnsiString) : Integer
6689: Function IsDriveReady( Drive : Char) : Boolean
6690: Function IsFile( const FileName : AnsiString) : Boolean
6691: Function IsFileArchive( const S : AnsiString) : Integer
6692: Function IsFileHidden( const S : AnsiString) : Integer
6693: Function IsFileReadOnly( const S : AnsiString) : Integer
6694: Function IsFileSystem( const S : AnsiString) : Integer
6695: Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6696: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6697: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6698: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6699: Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6700: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6701: Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6702: Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6703: Function ValidDrive( Drive : Char) : Boolean
6704: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6705:
6706: //*****unit uPSI_Jc1LANMan;*****
6707: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;  
const PasswordNeverExpires : Boolean) : Boolean
6708: Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;  
const PasswordNeverExpires : Boolean) : Boolean
6709: Function DeleteAccount( const Servername, Username : string) : Boolean
6710: Function DeleteLocalAccount( const Username : string) : Boolean
6711: Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6712: Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6713: Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6714: Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6715: Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6716: Function LocalGroupExists( const Group : string) : Boolean
6717: Function GlobalGroupExists( const Server, Group : string) : Boolean
6718: Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6719: Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6720: Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6721: Function IsLocalAccount( const AccountName : string) : Boolean
6722: Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6723: Function GetRandomString( NumChar : cardinal) : string
6724:
6725: //*****unit uPSI_cUtils;*****
6726: Types('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )'
6727: Function cIsWinNT : boolean
6728: Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,  
Multitasking:Boolean;
6729: Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6730: Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;  
CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6731: Function cGetShortName( const FileName : string) : string
6732: Procedure cShowError( Msg : string)
6733: Function cCommaStrToStr( s : string; formatstr : string) : string
6734: Function cIncludeQuoteIfSpaces( s : string) : string
6735: Function cIncludeQuoteIfNeeded( s : string) : string
6736: Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6737: Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6738: Function cBuildFilter( var value : string; const FLTStyle : TFLTSET) : boolean;
6739: Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6740: Function cCodeInstoStr( s : string) : string
6741: Function cStrtoCodeIns( s : string) : string
6742: Procedure cAttrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6743: Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6744: Procedure cStrtoPoint( var pt : TPoint; value : string)
6745: Function cPointtoStr( const pt : TPoint) : string
6746: Function cListtoStr( const List : TStrings) : string
6747: Function ListtoStr( const List : TStrings) : string
6748: Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6749: Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)

```

```

6750: Function cGetFileType( const FileName : string ) : TUnitType
6751: Function cGetExTyp( const FileName : string ) : TExUnitType
6752: Procedure cSetPath( Add : string; const UseOriginal : boolean )
6753: Function cExpandFileto( const FileName : string; const BasePath : string ) : string
6754: Function cFileSamePath( const FileName : string; const TestPath : string ) : boolean
6755: Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem )
6756: Function cGetLastPos( const SubStr : string; const S : string ) : integer
6757: Function cGenMakePath( FileName : String ) : String;
6758: Function cGenMakePath2( FileName : String ) : String
6759: Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean ) : String;
6760: Function cGetRealPath( BrokenFileName : String; Directory : String ) : String
6761: Function cCalcMod( Count : Integer ) : Integer
6762: Function cGetString( FileName : string ) : string
6763: Function cCheckChangeDir( var Dir : string ) : boolean
6764: Function cGetAssociatedProgram( const Extension:string; var Filename,Description: string ):boolean
6765: Function cIsNumeric( s : string ) : boolean
6766: Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6767: Function AttrToStr( const Attr : TSynHighlighterAttributes ) : string
6768: Function GetFileType( const FileName : string ) : TUnitType
6769: Function Atoi(const aStr: string): integer
6770: Function Itoa(const aint: integer): string
6771: Function Atof(const aStr: string): double';
6772: Function Atol(const aStr: string): longint';
6773:
6774:
6775: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6776: begin
6777:   FindClass ('TOBJECT'), 'EHTTP
6778:   FindClass ('TOBJECT'), 'EHTTPParser
6779: //AnsiCharSet', 'set of AnsiChar
6780: AnsiStringArray', 'array of AnsiString
6781: THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6782: THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6783: THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6784: +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6785: +'CustomMinVersion : Integer; end
6786: THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6787: +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6788: +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6789: +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6790: +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6791: +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6792: +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges, '
6793: +'hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSince'
6794: +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6795: +'nection, hntOrigin, hntKeepAlive )
6796: THTTPHeaderCodeName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6797: THTTPCustomHeader', 'record FieldName : AnsiString;FieldValue : '
6798: +' AnsiString; end
6799: //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6800: THTTPContentLengthEnum', '( hcNone, hc1ByteCount )
6801: THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6802: //PHTTPContentLength', '^THTTPContentLength // will not work
6803: THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6804: THTTPContentTypeEnum', '( hcNone, hctCustomParts, hctCustomStri'
6805: +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6806: +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6807: +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctAppli'
6808: +'ationCustom, hctAudioCustom, hctVideoCustom )
6809: THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6810: +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray; '
6811: +' CustomStr : AnsiString; end
6812: THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6813: THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek : '
6814: +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6815: +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6816: +'String; DateTime : TDateTime; Custom : AnsiString; end
6817: THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6818: THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6819: +'m; Custom : AnsiString; end
6820: THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6821: THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum; '
6822: +' Custom : AnsiString; end
6823: THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6824: THTTPAgeField', 'record Value : THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6825: THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6826: THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6827: +' , hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6828: THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6829: +' , hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6830: +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6831: THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6832: THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6833: +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6834: THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6835: THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6836: THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6837: +'FieldEnum; List : array of THTTPContentEncoding; end
6838: THTTPRetryAfterFieldEnum', '( harfNone, harfCustom, harfDate, harfSeconds )

```

```

6839: THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'  

6840: +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end  

6841: THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )  

6842: THTTPContentRangeField', 'record Value : THTTPContentRangeFieldD'  

6843: +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end  

6844: THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )  

6845: THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end  

6846: THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField  

6847: THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'  

6848: +'main : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '  

6849: +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'  

6850: +'CustomFieldArray; Custom : AnsiString; end  

6851: //THTTPSetCookieField', '^THTTPSetCookieField // will not work  

6852: THTTPSetCookieFieldArray', 'array of THTTPSetCookieField  

6853: THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )  

6854: THTTPCookiefieldEntry', 'record Name : AnsiString; Value : AnsiString; end  

6855: //THTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work  

6856: THTTPCookiefieldEntryArray', 'array of THTTPCookieFieldEntry  

6857: THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'  

6858: +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end  

6859: THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'  

6860: +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'  

6861: +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'  

6862: +' ; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end  

6863: THTTPCustomHeaders', 'array of THTTPCustomHeader  

6864: //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString  

6865: THTTPFixedHeaders','array[0..42] of AnsiString  

6866: THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'  

6867: +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )  

6868: THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end  

6869: THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end  

6870: THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;  

6871: +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'  

6872: +'kie:THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end  

6873: //THTTPRequestHeader', '^THTTPRequestHeader // will not work  

6874: THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'  

6875: +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end  

6876: THTTPResponseStartLineMessage', '( hs1mNone, hs1mCustom, hs1mOK)  

6877: THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '  

6878: +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end  

6879: THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'  

6880: +' ; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'  

6881: +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified : '  

6882: +' THTTPDateField; Age : THTTPAgeField; end  

6883: //THTTPResponseHeader', '^THTTPResponseHeader // will not work  

6884: THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'  

6885: +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end  

6886: Function HTTPMessageHasContent( const H : THTTPCommonHeaders ) : Boolean  

6887: Procedure InitHTTPRequest( var A : THTTPRequest )  

6888: Procedure InitHTTPResponse( var A : THTTPResponse )  

6889: Procedure ClearHTTPVersion( var A : THTTPVersion )  

6890: Procedure ClearHTTPContentLength( var A : THTTPContentLength )  

6891: Procedure ClearHTTPContentType( var A : THTTPContentType )  

6892: Procedure ClearHTTPDateField( var A : THTTPDateField )  

6893: Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding )  

6894: Procedure ClearHTTPConnectionField( var A : THTTPConnectionField )  

6895: Procedure ClearHTTPAgeField( var A : THTTPAgeField )  

6896: Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding )  

6897: Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField )  

6898: Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField )  

6899: Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField )  

6900: Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders )  

6901: //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders )  

6902: Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders )  

6903: Procedure ClearHTTPCookieField( var A : THTTPCookieField )  

6904: Procedure ClearHTTPMethod( var A : THTTPMethod )  

6905: Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine )  

6906: Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader )  

6907: Procedure ClearHTTPRequest( var A : THTTPRequest )  

6908: Procedure ClearHTTPResponseStartline( var A : THTTPResponseStartLine )  

6909: Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader )  

6910: Procedure ClearHTTPResponse( var A : THTTPResponse )  

6911: THTTPStringOption', '( hsNone )  

6912: THTTPStringOptions', 'set of THTTPStringOption  

6913: FindClass('TOBJECT'), 'TansiStringBuilder  

6914:  

6915: Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;  

6916: Procedure BuildStrHTTPContentLengthValue(const  

A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)  

6917: Procedure BuildStrHTTPContentLength(const A : THTTPContentlength;  

B:TAnsiStringBuilder;P:THTTPStringOptions)  

6918: Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const  

P:THTTPStringOptions)  

6919: Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const  

P:THTTPStringOptions)  

6920: Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const  

B : TAnsiStringBuilder; const P : THTTPStringOptions)  

6921: Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :  

THTTPStringOptions )

```

```

6922: Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
6923: P:THTTPStringOptions);
6924: Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
6925: TAnsiStringBuilder; const P : THTTPStringOptions)
6926: Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
6927: const P : THTTPStringOptions)
6928: Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
6929: const P : THTTPStringOptions)
6930: Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
6931: const P : THTTPStringOptions)
6932: Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
6933: const P : THTTPStringOptions)
6934: Procedure BuildStrHTTPPageField(const A:THTTPPageField;const B:TAnsiStringBuilder;const
6935: P:THTTPStringOptions);
6936: Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
6937: const P : THTTPStringOptions)
6938: Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
6939: B:TAnsiStringBuilder;const P:THTTPStringOptions)
6940: Procedure BuildStrHTTPProxyConnectionField(const A : THTTPProxyConnectionField; const B : TAnsiStringBuilder;
6941: const P : THTTPStringOptions)
6942: Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
6943: : THTTPStringOptions)
6944: Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
6945: P:THTTPStringOptions)
6946: Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
6947: : THTTPStringOptions)
6948: Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
6949: const P : THTTPStringOptions)
6950: Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
6951: : THTTPStringOptions)
6952: Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
6953: P:THTTPStringOptions);
6954: Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6955: Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6956: Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6957: Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6958: Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6959: Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6960: Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
6961: Domain:AnsiString;const Secure:Boolean; THTTPParserHeaderParseFunc,'Function (const HeaderName : THTT
6962: +PHeaderNameEnum; const HeaderPtr : __Pointer) : Boolean
6963: SIRegister_THTTPParser(CL);
6964: FindClass('TOBJECT','THTTPContentDecoder')
6965: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6966: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6967: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6968: +' crcsContentCRLF, crcsTrailer, crcsFinished )
6969: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6970: SIRegister_THTTPContentDecoder(CL);
6971: THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6972: FindClass('TOBJECT','THTTPContentReader')
6973: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6974: THTTPContentReaderLogEvent', 'Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
6975: LogLevel:Int;
6976: SIRegister_THTTPContentReader(CL);
6977: THTTPContentWriterMechanism',(hctmEvent, hctmString, hctmStream, hctmFile )
6978: FindClass('TOBJECT','THTTPContentWriter')
6979: THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6980: SIRegister_THTTPContentWriter(CL);
6981: Procedure SelfTestcHTTPUtils
6982: end;
6983:
6984: (*-----*)
6985: procedure SIRegister_cTLSUtils(CL: TPPascalCompiler);
6986: begin
6987: 'TLSLibraryVersion','String '1.00
6988: 'TLSerror_None','LongInt'( 0 );
6989: 'TLSerror_InvalidBuffer','LongInt'( 1 );
6990: 'TLSerror_InvalidParameter','LongInt'( 2 );
6991: 'TLSerror_InvalidCertificate','LongInt'( 3 );
6992: 'TLSerror_InvalidState','LongInt'( 4 );
6993: 'TLSerror_DecodeError','LongInt'( 5 );
6994: 'TLSerror_BadProtocol','LongInt'( 6 );
6995: Function TLSErrorMessage( const TLSerror : Integer) : String

```

```

6986: SIRRegister_ETLSError(CL);
6987: TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6988: PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6989: Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6990: Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6991: Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6992: Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6993: Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6994: Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6995: Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6996: Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6997: Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6998: Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6999: Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
7000: Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
7001: Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
7002: Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
7003: Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
7004: Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
7005: Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
7006: PTLSRandom', '^PTLSRandom // will not work
7007: Procedure InitTLSRandom( var Random : PTLSRandom)
7008: Function TLSRandomToStr( const Random : PTLSRandom) : AnsiString
7009: 'TLSSESSIONIDMaxLen','LongInt'( 32);
7010: Procedure InitTLSSessionID( var SessionID : TLSSessionID; const A : AnsiString)
7011: Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TLSSessionID):Int;
7012: Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TLSSessionID):Int;
7013: TTLSignatureAndHashAlgorithm', 'record Hash : TTLSSignatureAlgorithm'
7014: +' ; Signature : TTLSSignatureAlgorithm; end
7015: // TTLSignatureAndHashAlgorithm', '^TTLSignatureAndHashAlgorithm'// will not work
7016: TTLSignatureAndHashAlgorithmArray', 'array of TTLSignatureAndHashAlgorithm
7017: TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
7018: +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )'
7019: TTLSMACAlgorithm', '( tlsmNone, tlsmNULL, tlsm HMAC_MD5, tlsm'
7020: +' HMAC_SHA1, tlsm HMAC_SHA256, tlsm HMAC_SHA384, tlsm HMAC SHA512 )
7021: TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
7022: +'nteger; Supported : Boolean; end
7023: PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
7024: 'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
7025: TTLSPRFAlgorithm', '( tlspaSHA256 )
7026: Function tlsp_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
7027: Function tlsp_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
7028: Function tlsp_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
7029: Function tlsp_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
7030: Function tlsp_OPRF( const Secret, ALLabel, Seed : AnsiString; const Size : Integer) : AnsiString
7031: Function tlsp12PRF_SHA256( const Secret, ALLabel, Seed : AnsiString; const Size : Integer) : AnsiString
7032: Function tlsp12PRF_SHA512( const Secret, ALLabel, Seed : AnsiString; const Size : Integer) : AnsiString
7033: Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALLabel,Seed:AString;const
Size:Int):AString;
7034: Function tlsp10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
Size:Int):AnsiString
7035: Function tlsp12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
7036: Function tlsp12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
7037: Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : AnsiString; const Size : Integer) : AnsiString
7038: Function tlsp10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
7039: Function tlsp12SHA256MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString):AnsiString;
7040: Function tlsp12SHA512MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString):AnsiString;
7041: Function TLSMasterSecret ( const ProtocolVersion : TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
ServerRandom:AnsiString) : AnsiString
7042: TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
7043: +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
7044: +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
7045: Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
7046: Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
7047: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
7048: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7049: Procedure SelfTestcTLSUtils
7050: end;
7051:
7052: (*-----*)
7053: procedure SIRRegister_Reversi(CL: TPSPascalCompiler);
7054: begin
7055: sPosData', 'record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
integer; disks : integer; mx : integer; my : integer; end
7056: // pBoard', '^tBoard // will not work
7057: Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
7058: Function rCheckMove( color : byte; cx, cy : integer) : integer
7059: //Function rDoStep( data : pBoard) : word
7060: Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
7061: end;
7062:
7063: procedure SIRRegister_IWDBCommon(CL: TPSPascalCompiler);
7064: begin
7065: Function InEditMode( ADataset : TDataset) : Boolean

```

```

7066: Function CheckDataSource( ADataSource : TDataSource ) : Boolean;
7067: Function CheckDataSource1( ADataSource:TDataSource;const AFieldName:string;var VField:TField ):boolean;
7068: Function GetFieldText( AField : TField ) : String
7069: end;
7070:
7071: procedure SIRegister_SortGrid(CL: TPSPPascalCompiler);
7072: begin
7073:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7074:   TMyPrintRange', '( prAll, prSelected )
7075:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7076:     +'ded, ssDateTime, ssTime, ssCustom )
7077:   TSortDirection', '( sdAscending, sdDescending )
7078:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7079:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7080:     +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7081:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7082:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7083:   SIRegister_TSortOptions(CL);
7084:   SIRegister_TPrintOptions(CL);
7085:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7086:   SIRegister_TSORTedList(CL);
7087:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7088:   TCellBevel', 'record Style: TCellBevelStyle;UpperLeftColor:TColor;LowerRightColor:TColor; end
7089:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7090:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7091:     +'Horz : TAlignment; AlignmentVert: TVertAlignment; Bevel:TCellBevel; HideText:Boolean; end
7092:   SIRegister_TFontSetting(CL);
7093:   SIRegister_TFontList(CL);
7094:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row : '
7095:     + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7096:   TSetFilterEvent', 'Procedure ( ARows : TStringList; var Accept : Boolean)
7097:   TSearchEvent', 'Procedure ( ARows : TStringList; var Accept : Boolean)
7098:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7099:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7100:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7101:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7102:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer'
7103:     +'r; var SortStyle : TSortStyle)
7104:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow : '
7105:     +'integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7106:   SIRegister_TSORTgrid(CL);
7107:   Function ExtendedCompare( const Str1, Str2 : String ) : Integer
7108:   Function NormalCompare( const Str1, Str2 : String ) : Integer
7109:   Function DateTimeCompare( const Str1, Str2 : String ) : Integer
7110:   Function NumericCompare( const Str1, Str2 : String ) : Integer
7111:   Function TimeCompare( const Str1, Str2 : String ) : Integer
7112: //Function Compare( Item1, Item2 : Pointer ) : Integer
7113: end;
7114:
7115: *****unit uPSI_BoldUtils;*****
7116: Procedure IBAalloc( var P, OldSize, NewSize : Integer)
7117: Procedure IBEerror( ErrMess : TIBClientError; const Args : array of const)
7118: Procedure IB DataBaseError
7119: Function StatusVector : PISC_STATUS
7120: Function StatusVectorArray : PStatusVector
7121: Function CheckStatusVector( ErrorCode : array of ISC_STATUS ) : Boolean
7122: Function StatusVectorAsText : string
7123: Procedure SetIB DataBaseErrorMessages( Value : TIB DataBaseErrorMessages)
7124: Function GetIB DataBaseErrorMessages : TIB DataBaseErrorMessages
7125:
7126:
7127: //*****unit uPSI_BoldUtils;*****
7128: Function CharCount( c : char; const s : string ) : integer
7129: Function BoldNamesEqual( const name1, name2 : string ) : Boolean
7130: Procedure BoldAppendToStrings(strings: TStringList; const aString: string; const ForceNewLine:Boolean)
7131: Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7132: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string ) : string
7133: Function BoldTrim( const S : string ) : string
7134: Function BoldIsPrefix( const S, Prefix : string ) : Boolean
7135: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7136: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7137: Function BoldAnsiEqual( const S1, S2 : string ) : Boolean
7138: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7139: Function BoldCaseIndependentPos( const Substr, S : string ) : Integer
7140: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStringList)
7141: Function CapitalisedToSpaced( Capitalised : String ) : String
7142: Function SpacedToCapitalised( Spaced : String ) : String
7143: Function BooleanToString( BoolValue : Boolean ) : String
7144: Function StringToBoolean( StrValue : String ) : Boolean
7145: Function GetUpperLimitForMultiplicity( const Multiplicity : String ) : Integer
7146: Function GetLowerLimitForMultiplicity( const Multiplicity : String ) : Integer
7147: Function StringListToVarArray( List : TStringList ) : variant
7148: Function IsLocalMachine( const MachineName : WideString ) : Boolean
7149: Function GetComputerNameStr : string
7150: Function TimeStampComp( const Time1, Time2 : TTimeStamp ) : Integer
7151: Function BoldStrToDateFmt( const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7152: Function BoldDateToStrFmt( const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7153: Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7154: Function BoldParseFormattedDate(const value:String;const formats:array of string; var Date:TDateTime):Boolean;

```

```

7155: Procedure EnsureTrailing( var Str : String; ch : char)
7156: Function BoldDirectoryExists( const Name : string) : Boolean
7157: Function BoldForceDirectories( Dir : string) : Boolean
7158: Function BoldRootRegistryKey : string
7159: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7160: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7161: Function LogicalAnd( A, B : Integer) : Boolean
7162: record TByHandleFileInformation dwFileAttributes : DWORD;
7163:   +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7164:   +' : TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSize'
7165:   +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7166: Function GetfileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation) :BOOL;
7167: Function IsFirstInstance : Boolean
7168: Procedure ActivateFirst( AString : PChar)
7169: Procedure ActivateFirstCommandLine
7170: function MakeAckPkt(const BlockNumber: Word) : string;
7171: procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APoRt:Int;const ErrNumber:Word;ErrStr:string;
7172: procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7173: procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APoRt: Integer; E: Exception); overload;
7174: procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7175: function IdStrToWord(const Value: String): Word;
7176: function IdWordToStr(const Value: Word): WordStr;
7177: Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7178: Function CPUFeatures : TCPUCPUFeatures
7179:
7180: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7181: begin
7182:   AddTypeS('TXRTLBIndex', 'Integer'
7183:   Function XRTLSwapbits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBIndex) : Cardinal
7184:   Function XRTLBBitTest( Data : Cardinal; BitIndex : TXRTLBIndex) : Boolean
7185:   Function XRTLBBitSet( Data : Cardinal; BitIndex : TXRTLBIndex) : Cardinal
7186:   Function XRTLBBitReset( Data : Cardinal; BitIndex : TXRTLBIndex) : Cardinal
7187:   Function XRTLBBitComplement( Data : Cardinal; BitIndex : TXRTLBIndex) : Cardinal
7188:   Function XRTLSwapHiLo16( X : Word) : Word
7189:   Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7190:   Function XRTLSwapHiLo64( X : Int64) : Int64
7191:   Function XRTLROL32( A, S : Cardinal) : Cardinal
7192:   Function XRTLROLR32( A, S : Cardinal) : Cardinal
7193:   Function XRTLROL16( A : Word; S : Cardinal) : Word
7194:   Function XRTLROLR16( A : Word; S : Cardinal) : Word
7195:   Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7196:   Function XRTLROLR8( A : Byte; S : Cardinal) : Byte
7197: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7198: //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7199: Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7200: Function XRTLPopulation( A : Cardinal) : Cardinal
7201: end;
7202:
7203: Function XRTLURLDecode( const ASrc : WideString) : WideString
7204: Function XRTLURLEncode( const ASrc : WideString) : WideString
7205: Function XRTLURINormalize( const AURI : WideString) : WideString
7206: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
    VPassword : WideString)
7207: Function XRTLEExtractLongPathName(APath: string) : string;
7208:
7209: procedure SIRegister_cfFundamentUtils(CL: TPSPascalCompiler);
7210: begin
7211:   Int8', 'ShortInt
7212:   AddTypeS('Int16', 'SmallInt
7213:   Int32', 'LongInt
7214:   UInt8', 'Byte
7215:   UInt16', 'Word
7216:   UInt32', 'LongWord
7217:   UInt64', 'Int64
7218:   Word8', 'UInt8
7219:   Word16', 'UInt16
7220:   Word32', 'UInt32
7221:   Word64', 'UInt64
7222:   LargeInt', 'Int64
7223:   NativeInt', 'Integer
7224:   AddTypeS('NativeUInt', 'Cardinal
7225:   Const('BitsPerByte','LongInt'( 8);
7226:   Const('BitsPerWord','LongInt'( 16);
7227:   Const('BitsPerLongWord','LongInt'( 32);
7228: //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7229: //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7230: Function MinI( const A, B : Integer) : Integer
7231: Function MaxI( const A, B : Integer) : Integer
7232: Function MinC( const A, B : Cardinal) : Cardinal
7233: Function MaxC( const A, B : Cardinal) : Cardinal
7234: Function SumClipI( const A, I : Integer) : Integer
7235: Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7236: Function InByteRange( const A : Int64) : Boolean
7237: Function InWordRange( const A : Int64) : Boolean
7238: Function InLongWordRange( const A : Int64) : Boolean
7239: Function InShortIntRange( const A : Int64) : Boolean
7240: Function InSmallIntRange( const A : Int64) : Boolean
7241: Function InLongIntRange( const A : Int64) : Boolean
7242:   AddTypeS('Bool8', 'ByteBool

```

```

7243: AddTypeS('Bool16', 'WordBool
7244: AddTypeS('Bool32', 'LongBool
7245: AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7246: AddTypeS('TCompareResultSet', 'set of TCompareResult
7247: Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7248: Const ('MinSingle','Single').setExtended( 1.5E-45);
7249: Const ('MaxSingle','Single').setExtended( 3.4E+38);
7250: Const ('MinDouble','Double').setExtended( 5.0E-324);
7251: Const ('MaxDouble','Double').setExtended( 1.7E+308);
7252: Const ('MinExtended','Extended').setExtended(3.4E-4932);
7253: Const ('MaxExtended','Extended').setExtended(1.1E+4932);
7254: Const ('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7255: Const ('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7256: Function MinF( const A, B : Float ) : Float
7257: Function MaxF( const A, B : Float ) : Float
7258: Function ClipF( const Value : Float; const Low, High : Float ) : Float
7259: Function InSingleRange( const A : Float ) : Boolean
7260: Function InDoubleRange( const A : Float ) : Boolean
7261: Function InCurrencyRange( const A : Float ) : Boolean;
7262: Function InCurrencyRangel( const A : Int64 ) : Boolean;
7263: Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7264: Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7265: Function FloatIsInfinity( const A : Extended ) : Boolean
7266: Function FloatIsNaN( const A : Extended ) : Boolean
7267: Const ('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7268: Const ('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7269: Const ('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7270: Const ('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7271: Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7272: Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7273: Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7274: Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult
7275: Const ('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7276: Const ('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7277: Const ('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7278: Const ('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7279: Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double ) : Boolean
7280: Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7281: Function cClearBit( const Value, BitIndex : LongWord ) : LongWord
7282: Function cSetBit( const Value, BitIndex : LongWord ) : LongWord
7283: Function cIsBitSet( const Value, BitIndex : LongWord ) : Boolean
7284: Function cToggleBit( const Value, BitIndex : LongWord ) : LongWord
7285: Function cIsHighBitSet( const Value : LongWord ) : Boolean
7286: Function SetBitScanForward( const Value : LongWord ) : Integer;
7287: Function SetBitScanForward1( const Value : BitIndex : LongWord ) : Integer;
7288: Function SetBitScanReverse( const Value : LongWord ) : Integer;
7289: Function SetBitScanReverse1( const Value : BitIndex : LongWord ) : Integer;
7290: Function ClearBitScanForward( const Value : LongWord ) : Integer;
7291: Function ClearBitScanForward1( const Value, BitIndex : LongWord ) : Integer;
7292: Function ClearBitScanReverse( const Value : LongWord ) : Integer;
7293: Function ClearBitScanReverse1( const Value, BitIndex : LongWord ) : Integer;
7294: Function cReversebits( const Value : LongWord ) : LongWord;
7295: Function cReverseBits1( const Value : LongWord; const BitCount : Integer ) : LongWord;
7296: Function cSwapEndian( const Value : LongWord ) : LongWord
7297: Function cTwosComplement( const Value : LongWord ) : LongWord
7298: Function RotateLeftBits16( const Value : Word; const Bits : Byte ) : Word
7299: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte ) : LongWord
7300: Function RotateRightBits16( const Value : Word; const Bits : Byte ) : Word
7301: Function RotateRightBits32( const Value : LongWord; const Bits : Byte ) : LongWord
7302: Function cBitCount( const Value : LongWord ) : LongWord
7303: Function cIsPowerOfTwo( const Value : LongWord ) : Boolean
7304: Function LowBitMask( const HighBitIndex : LongWord ) : LongWord
7305: Function HighBitMask( const LowBitIndex : LongWord ) : LongWord
7306: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord ) : LongWord
7307: Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord ) : LongWord
7308: Function ClearBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord ) : LongWord
7309: Function ToggleBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord ) : LongWord
7310: Function IsBitRangeSet( const Value: LongWord; const LowBitIndex, HighBitIndex : LongWord ) : Boolean
7311: Function IsBitRangeClear( const Value: LongWord; const LowBitIndex, HighBitIndex : LongWord ) : Boolean
7312: // AddTypeS('CharSet', 'set of AnsiChar
7313: AddTypeS('CharSet', 'set of Char' //!!!
7314: AddTypeS('AnsiCharSet', 'TCharSet
7315: AddTypeS('ByteSet', 'set of Byte
7316: AddTypeS('AnsiChar', 'Char
7317: // Function AsCharSet( const C : array of AnsiChar ) : CharSet
7318: Function AsByteSet( const C : array of Byte ) : ByteSet
7319: Procedure ComplementChar( var C : CharSet; const Ch : Char)
7320: Procedure ClearCharSet( var C : CharSet)
7321: Procedure FillCharSet( var C : CharSet)
7322: procedure FillCharSearchRec; // with 0
7323: Procedure ComplementCharSet( var C : CharSet)
7324: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7325: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7326: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7327: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7328: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7329: Function IsSubSet( const A, B : CharSet ) : Boolean
7330: //Function IsEqual( const A, B : CharSet ) : Boolean
7331: Function IsEqualCharset( const A, B : CharSet ) : Boolean

```

```

7332: Function IsEqual( const D1, D2 : TDateTime ) : Boolean;
7333: Function IsEmpty( const C : CharSet ) : Boolean
7334: Function IsEmptyCharset( const C : CharSet ) : Boolean
7335: Function IsComplete( const C : CharSet ) : Boolean
7336: Function cCharCount( const C : CharSet ) : Integer
7337: Procedure ConvertCaseInsensitive( var C : CharSet )
7338: Function CaseInsensitiveCharSet( const C : CharSet ) : CharSet
7339: Function IntRangeLength( const Low, High : Integer ) : Int64
7340: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer ) : Boolean
7341: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer ) : Boolean
7342: Function IntRangeHasElement( const Low, High, Element : Integer ) : Boolean
7343: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer ) : Boolean
7344: Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7345: Function CardRangeLength( const Low, High : Cardinal ) : Int64
7346: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal ) : Boolean
7347: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal ) : Boolean
7348: Function CardRangeHasElement( const Low, High, Element : Cardinal ) : Boolean
7349: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal ) : Boolean
7350: Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Bool;
7351: AddTypeS('UnicodeChar', 'WideChar'
7352: Function Compare( const I1, I2 : Boolean ) : TCompareResult;
7353: Function CompareI( const I1, I2 : Integer ) : TCompareResult;
7354: Function Compare2( const I1, I2 : Int64 ) : TCompareResult;
7355: Function Compare3( const I1, I2 : Extended ) : TCompareResult;
7356: Function CompareA( const I1, I2 : AnsiString ) : TCompareResult
7357: Function CompareW( const I1, I2 : WideString ) : TCompareResult
7358: Function cSgn( const A : LongInt ) : Integer;
7359: Function cSgn1( const A : Int64 ) : Integer;
7360: Function cSgn2( const A : Extended ) : Integer;
7361: AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )'
7362: Function AnsiCharToInt( const A : AnsiChar ) : Integer
7363: Function WideCharToInt( const A : WideChar ) : Integer
7364: Function CharToInt( const A : Char ) : Integer
7365: Function IntToAnsiChar( const A : Integer ) : AnsiChar
7366: Function IntToWideChar( const A : Integer ) : WideChar
7367: Function IntToChar( const A : Integer ) : Char
7368: Function IsHexAnsiChar( const Ch : AnsiChar ) : Boolean
7369: Function IsHexWideChar( const Ch : WideChar ) : Boolean
7370: Function IsHexChar( const Ch : Char ) : Boolean
7371: Function HexAnsiCharToInt( const A : AnsiChar ) : Integer
7372: Function HexWideCharToInt( const A : WideChar ) : Integer
7373: Function HexCharToInt( const A : Char ) : Integer
7374: Function IntToUpperHexAnsiChar( const A : Integer ) : AnsiChar
7375: Function IntToUpperHexWideChar( const A : Integer ) : WideChar
7376: Function IntToUpperHexChar( const A : Integer ) : Char
7377: Function IntToLowerHexAnsiChar( const A : Integer ) : AnsiChar
7378: Function IntToLowerHexWideChar( const A : Integer ) : WideChar
7379: Function IntToLowerHexChar( const A : Integer ) : Char
7380: Function IntToStringA( const A : Int64 ) : AnsiString
7381: Function IntToStringW( const A : Int64 ) : WideString
7382: Function IntToString( const A : Int64 ) : String
7383: Function UIntToStringA( const A : NativeUInt ) : AnsiString
7384: Function UIntToStringW( const A : NativeUInt ) : WideString
7385: Function UIntToString( const A : NativeUInt ) : String
7386: Function LongWordToStrA( const A : LongWord; const Digits : Integer ) : AnsiString
7387: Function LongWordToStrW( const A : LongWord; const Digits : Integer ) : WideString
7388: Function LongWordToStrU( const A : LongWord; const Digits : Integer ) : UnicodeString
7389: Function LongWordToStr( const A : LongWord; const Digits : Integer ) : String
7390: Function LongWordToHexA( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7391: Function LongWordToHexW( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7392: Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7393: Function LongWordToOctA( const A : LongWord; const Digits : Integer ) : AnsiString
7394: Function LongWordToOctW( const A : LongWord; const Digits : Integer ) : WideString
7395: Function LongWordToOct( const A : LongWord; const Digits : Integer ) : String
7396: Function LongWordToBinA( const A : LongWord; const Digits : Integer ) : AnsiString
7397: Function LongWordToBinW( const A : LongWord; const Digits : Integer ) : WideString
7398: Function LongWordToBin( const A : LongWord; const Digits : Integer ) : String
7399: Function TryStringToInt64A( const S : AnsiString; out A : Int64 ) : Boolean
7400: Function TryStringToInt64W( const S : WideString; out A : Int64 ) : Boolean
7401: Function TryStringToInt64( const S : String; out A : Int64 ) : Boolean
7402: Function StringToInt64DefA( const S : AnsiString; const Default : Int64 ) : Int64
7403: Function StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7404: Function StringToInt64Def( const S : String; const Default : Int64 ) : Int64
7405: Function StringToInt64A( const S : AnsiString ) : Int64
7406: Function StringToInt64W( const S : WideString ) : Int64
7407: Function StringToInt64( const S : String ) : Int64
7408: Function TryStringToIntA( const S : AnsiString; out A : Integer ) : Boolean
7409: Function TryStringToIntW( const S : WideString; out A : Integer ) : Boolean
7410: Function TryStringToInt( const S : String; out A : Integer ) : Boolean
7411: Function StringToIntDefA( const S : AnsiString; const Default : Integer ) : Integer
7412: Function StringToIntDefW( const S : WideString; const Default : Integer ) : Integer
7413: Function StringToIntDef( const S : String; const Default : Integer ) : Integer
7414: Function StringToIntA( const S : AnsiString ) : Integer
7415: Function StringToIntW( const S : WideString ) : Integer
7416: Function StringToInt( const S : String ) : Integer
7417: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7418: Function TryStringToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7419: Function TryStringToLongWord( const S : String; out A : LongWord ) : Boolean
7420: Function StringToLongWordA( const S : AnsiString ) : LongWord

```

```

7421: Function StringToLongWordW( const S : WideString ) : LongWord
7422: Function StringToLongWord( const S : String ) : LongWord
7423: Function HexToIntA( const S : AnsiString ) : NativeUInt
7424: Function HexToIntW( const S : WideString ) : NativeUInt
7425: Function HexToInt( const S : String ) : NativeUInt
7426: Function TryHexToIntWordA( const S : AnsiString; out A : LongWord ) : Boolean
7427: Function TryHexToIntWordW( const S : WideString; out A : LongWord ) : Boolean
7428: Function TryHexToIntWord( const S : String; out A : LongWord ) : Boolean
7429: Function HexToLongWordA( const S : AnsiString ) : LongWord
7430: Function HexToLongWordW( const S : WideString ) : LongWord
7431: Function HexToLongWord( const S : String ) : LongWord
7432: Function TryOctToIntWordA( const S : AnsiString; out A : LongWord ) : Boolean
7433: Function TryOctToIntWordW( const S : WideString; out A : LongWord ) : Boolean
7434: Function TryOctToIntWord( const S : String; out A : LongWord ) : Boolean
7435: Function OctToIntWordA( const S : AnsiString ) : LongWord
7436: Function OctToIntWordW( const S : WideString ) : LongWord
7437: Function OctToIntWord( const S : String ) : LongWord
7438: Function TryBinToIntWordA( const S : AnsiString; out A : LongWord ) : Boolean
7439: Function TryBinToIntWordW( const S : WideString; out A : LongWord ) : Boolean
7440: Function TryBinToIntWord( const S : String; out A : LongWord ) : Boolean
7441: Function BinToIntWordA( const S : AnsiString ) : LongWord
7442: Function BinToIntWordW( const S : WideString ) : LongWord
7443: Function BinToIntWord( const S : String ) : LongWord
7444: Function FloatToStringA( const A : Extended ) : AnsiString
7445: Function FloatToStringW( const A : Extended ) : WideString
7446: Function FloatToString( const A : Extended ) : String
7447: Function TryStringToFloatA( const A : AnsiString; out B : Extended ) : Boolean
7448: Function TryStringToFloatW( const A : WideString; out B : Extended ) : Boolean
7449: Function TryStringToFloat( const A : String; out B : Extended ) : Boolean
7450: Function StringToFloatA( const A : AnsiString ) : Extended
7451: Function StringToFloatW( const A : WideString ) : Extended
7452: Function StringToFloat( const A : String ) : Extended
7453: Function StringToFloatDefA( const A : AnsiString; const Default : Extended ) : Extended
7454: Function StringToFloatDefW( const A : WideString; const Default : Extended ) : Extended
7455: Function StringToFloatDef( const A : String; const Default : Extended ) : Extended
7456: Function EncodeBase64( const S, Alphabet:AnsiString; const Pad:Boolean; const PadMultiple:Integer; const PadChar: AnsiChar ) : AnsiString
7457: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet ) : AnsiString
7458: unit uPSI_cfFundamentals;
7459: Const ('b64_MIMEBase64','Str').String('ABCDEFIGHJKLMNOPQRSTUVWXYZabcdeffghijklmnopqrstuvwxyz0123456789+/');
7460: Const ('b64_UUEncode','String').String('!'#$%&'+,-./0123456789;:<=>?ABCDEFIGHJKLMNOPQRSTUVWXYZ[\]^_');
7461: Const ('b64_XXEncode','String').String('+-0123456789ABCDEFIGHJKLMNOPQRSTUVWXYZabcdeffghijklmnopqrstuvwxyz');
7462: Const ('CCHARSET','String'64_XXEncode);
7463: Const ('CHEXSET','String'0123456789ABCDEF
7464: Const ('HEXDIGITS','String'0123456789ABCDEF
7465: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7466: Const ('DIGITSET','String'0123456789
7467: Const ('LETTERSET','String'ABCDEFIGHJKLMNOPQRSTUVWXYZ'
7468: Const ('DIGITSET2','TCharset').SetSet('0123456789'
7469: Const ('LETTERSET2','TCharset').SetSet('ABCDEFIGHJKLMNOPQRSTUVWXYZ'
7470: Const ('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7471: Const ('NUMBERSET','TCharset').SetSet('0123456789)';
7472: Const ('NUMBERS','String'0123456789);
7473: Const ('LETTERS','String'ABCDEFIGHJKLMNOPQRSTUVWXYZ');
7474: Const ('NUMBLETTSET','String').SetString('0123456789ABCDEFIGHJKLMNOPQRSTUVWXYZ');
7475: Const ('NUMBLETTSET','TCharset').SetSet('0123456789ABCDEFIGHJKLMNOPQRSTUVWXYZ');
7476: Function CharSetToStr( const C : CharSet ) : AnsiString
7477: Function StrToCharSet( const S : AnsiString ) : CharSet
7478: Function MIMEBase64Decode( const S : AnsiString ) : AnsiString
7479: Function MIMEBase64Encode( const S : AnsiString ) : AnsiString
7480: Function UUDecode( const S : AnsiString ) : AnsiString
7481: Function XXDecode( const S : AnsiString ) : AnsiString
7482: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean ) : AnsiString
7483: Function InterfaceToStrA( const I : IInterface ) : AnsiString
7484: Function InterfaceToStrW( const I : IInterface ) : WideString
7485: Function InterfaceToStr( const I : IInterface ) : String
7486: Function ObjectClassName( const O : TObject ) : String
7487: Function ClassClassName( const C : TClass ) : String
7488: Function ObjectToStr( const O : TObject ) : String
7489: Function ObjectToString( const O : TObject ) : String
7490: Function CharSetToStr( const C : CharSet ) : AnsiString
7491: Function StrToCharSet( const S : AnsiString ) : CharSet
7492: Function HashStrA( const S : AnsiString; const Index : Integer; const Count : Integer; const AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7493: Function HashStrW( const S:WideString;const Index:Integer;const Count:Integer;const AsciiCaseSensitive:Boolean; const Slots:LongWord ) : LongWord
7494: Function HashStr( const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7495: Function HashInteger( const I : Integer; const Slots : LongWord ) : LongWord
7496: Function HashLongWord( const I : LongWord; const Slots : LongWord ) : LongWord
7497: Const ('Bytes1KB','LongInt'( 1024 );
7498: SIRegister_IInterface(CL);
7499: Procedure SelfTestCFundamentals
7500:
7501: Function CreateSchedule : IJclSchedule
7502: Function NullStamp : TTimeStamp
7503: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
7504: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
7505: Function IsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean

```

```

7506:
7507: procedure SIRегистер_uwinplot(CL: TPSPPascalCompiler);
7508: begin
7509:   AddTypeS('TFunc', 'function(X : Float) : Float;');
7510:   Function InitGraphics( Width, Height : Integer ) : Boolean;
7511:   Procedure SetWindow( Canvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean );
7512:   Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float );
7513:   Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float );
7514:   Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float );
7515:   Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float );
7516:   Procedure SetGraphTitle( Title : String );
7517:   Procedure SetOxTitle( Title : String );
7518:   Procedure SetOyTitle( Title : String );
7519:   Function GetGraphTitle : String;
7520:   Function GetOxTitle : String;
7521:   Function GetOyTitle : String;
7522:   Procedure PlotOxAxis( Canvas : TCanvas );
7523:   Procedure PlotOyAxis( Canvas : TCanvas );
7524:   Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid );
7525:   Procedure WriteGraphTitle( Canvas : TCanvas );
7526:   Function SetMaxCurv( NCurv : Byte ) : Boolean;
7527:   Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor );
7528:   Procedure SetLineParam( CurvIndex: Integer; Style : TPenStyle; Width : Integer; Color : TColor );
7529:   Procedure SetCurvLegend( CurvIndex : Integer; Legend : String );
7530:   Procedure SetCurvStep( CurvIndex, Step : Integer );
7531:   Function GetMaxCurv : Byte;
7532:   Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor );
7533:   Procedure GetLineParam( CurvIndex:Integer;var Style:TPenStyle;var Width:Integer;var Color:TColor );
7534:   Function GetCurvLegend( CurvIndex : Integer ) : String;
7535:   Function GetCurvStep( CurvIndex : Integer ) : Integer;
7536:   Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer );
7537:   Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer );
7538:   Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer);
7539:   Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer );
7540:   Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean );
7541:   Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix );
7542:   Function Xpixel( X : Float ) : Integer;
7543:   Function Ypixel( Y : Float ) : Integer;
7544:   Function Xuser( X : Integer ) : Float;
7545:   Function Yuser( Y : Integer ) : Float;
7546: end;
7547:
7548: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector );
7549: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector );
7550: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector );
7551: Procedure FFT_Integer_Cleanup;
7552: Procedure CalcFrequency(NumSamples, FrequencyIndex: Integer; InArray: TCompVector; var FT : Complex );
7553: //unit uPSI_JclStreams;
7554: Function StreamSeek(Stream: TStream; const Offset: Int64; const Origin : TSeekOrigin) : Int64;
7555: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer ) : Int64;
7556: Function CompareStreams( A, B : TStream; BufferSize : Integer ) : Boolean;
7557: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer ) : Boolean;
7558:
7559: procedure SIRегистер_FmxUtils(CL: TPSPPascalCompiler);
7560: begin
7561:   FindClass('TOBJECT','EInvalidDest');
7562:   FindClass('TOBJECT','EFCantMove');
7563:   Procedure fmxCopyFile( const FileName, DestName : string );
7564:   Procedure fmxMoveFile( const FileName, DestName : string );
7565:   Function fmxGetFileSize( const FileName : string ) : LongInt;
7566:   Function fmxFileDialogTime( const FileName : string ) : TDateTime;
7567:   Function fmxHasAttr( const FileName : string; Attr : Word ) : Boolean;
7568:   Function fmxExecuteFile( const FileName, Params, DefaultDir: string; ShowCmd : Integer ) : THandle;
7569: end;
7570:
7571: procedure SIRегистер_FindFileIter(CL: TPSPPascalCompiler);
7572: begin
7573:   SIRегистер_IFindFileIterator(CL);
7574:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7575: end;
7576:
7577: procedure SIRегистер_PCharUtils(CL: TPSPPascalCompiler);
7578: begin
7579:   Function SkipWhite( cp : PChar ) : PChar;
7580:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string ) : PChar;
7581:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string ) : PChar;
7582:   Function ReadIdent( cp : PChar; var ident : string ) : PChar;
7583: end;
7584:
7585: procedure SIRегистер_JclStrHashMap(CL: TPSPPascalCompiler);
7586: begin
7587:   SIRегистер_TStringHashMapTraits(CL);
7588:   Function CaseSensitiveTraits : TStringHashMapTraits;
7589:   Function CaseInsensitiveTraits : TStringHashMapTraits;
7590:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod';
7591:     +'e; Right : PHashNode; end';
7592:   //PHashArray', 'THashArray // will not work
7593:   SIRегистер_TStringHashMap(CL);
7594:   THashValue', 'Cardinal;

```

```

7595: Function StrHash( const s : string) : THashValue
7596: Function TextHash( const s : string) : THashValue
7597: Function DataHash( var AValue, ASize : Cardinal) : THashValue
7598: Function Iterate_FreeObjects( AUUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7599: Function Iterate_Dispose( AUUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7600: Function Iterate_FreeMem( AUUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7601: SIRegister_TCaseSensitiveTraits(CL);
7602: SIRegister_TCaseInsensitiveTraits(CL);
7603:
7604:
7605: //*****unit uPSI_umath;
7606: Function uExpo( X : Float) : Float
7607: Function uExp2( X : Float) : Float
7608: Function uExp10( X : Float) : Float
7609: Function uLog( X : Float) : Float
7610: Function uLog2( X : Float) : Float
7611: Function uLog10( X : Float) : Float
7612: Function uLogA( X, A : Float) : Float
7613: Function uIntPower( X : Float; N : Integer) : Float
7614: Function uPower( X, Y : Float) : Float
7615: Function SgnGamma( X : Float) : Integer
7616: Function Stirling( X : Float) : Float
7617: Function StirLog( X : Float) : Float
7618: Function Gamma( X : Float) : Float
7619: Function LnGamma( X : Float) : Float
7620: Function DiGamma( X : Float) : Float
7621: Function TriGamma( X : Float) : Float
7622: Function IGamma( X : Float) : Float
7623: Function JGamma( X : Float) : Float
7624: Function InvGamma( X : Float) : Float
7625: Function Erf( X : Float) : Float
7626: Function Erfc( X : Float) : Float
7627: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7628: { Correlation coefficient between samples X and Y }
7629: function DBeta(A, B, X : Float) : Float;
7630: { Density of Beta distribution with parameters A and B }
7631: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7632: Function Beta(X, Y : Float) : Float
7633: Function Binomial( N, K : Integer) : Float
7634: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7635: Procedure Cholesky( A, L : TMMatrix; Lb, Ub : Integer)
7636: Procedure LU_Decom( A : TMMatrix; Lb, Ub : Integer)
7637: Procedure LU_Solve( A : TMMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7638: Function DNorm( X : Float) : Float
7639:
7640: function DGamma(A, B, X : Float) : Float;
7641: { Density of Gamma distribution with parameters A and B }
7642: function DKhi2(Nu : Integer; X : Float) : Float;
7643: { Density of Khi-2 distribution with Nu d.o.f. }
7644: function DStudent(Nu : Integer; X : Float) : Float;
7645: { Density of Student distribution with Nu d.o.f. }
7646: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7647: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7648: function IBeta(A, B, X : Float) : Float;
7649: { Incomplete Beta function}
7650: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7651:
7652: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7653: begin
7654: Procedure SetOptAlgo( Algo : TOptAlgo)
7655: procedure SetOptAlgo(Algo : TOptAlgo);
7656: { -----
7657: Sets the optimization algorithm according to Algo, which must be
7658: NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ }
7659:
7660: Function GetOptAlgo : TOptAlgo
7661: Procedure SetMaxParam( N : Byte)
7662: Function GetMaxParam : Byte
7663: Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7664: Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7665: Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7666: Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDervProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
    Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMMatrix)
7667: Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDervProc; X, Y, S : TVector; Lb, Ub : Integer;
    MaxIter:Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMMatrix)
7668: Procedure SetMCFile( FileName : String)
7669: Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMMatrix;
7670: Procedure WSImFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
    LastPar:Integer;V:TMMatrix);
7671: end;
7672:
7673: (*-----*)
7674: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7675: begin
7676: Procedure SaveSimplex( FileName : string)
7677: Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7678: end;
7679: (*-----*)
7680: procedure SIRegister uregtest(CL: TPSPascalCompiler);

```

```

7681: begin
7682:   Procedure RegTest(Y,Ycalc:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7683:   Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7684: end;
7685:
7686: procedure SIRegister_ustrings(CL: TPSPPascalCompiler);
7687: begin
7688:   Function LTrim( S : String ) : String
7689:   Function RTrim( S : String ) : String
7690:   Function uTrim( S : String ) : String
7691:   Function StrChar( N : Byte; C : Char ) : String
7692:   Function RFill( S : String; L : Byte ) : String
7693:   Function LFill( S : String; L : Byte ) : String
7694:   Function CFill( S : String; L : Byte ) : String
7695:   Function Replace( S : String; C1, C2 : Char ) : String
7696:   Function Extract( S : String; var Index : Byte; Delim : Char ) : String
7697:   Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte )
7698:   Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean )
7699:   Function FloatStr( X : Float ) : String
7700:   Function IntStr( N : LongInt ) : String
7701:   Function uCompStr( Z : Complex ) : String
7702: end;
7703:
7704: procedure SIRegister_uhyper(CL: TPSPPascalCompiler);
7705: begin
7706:   Function uSinh( X : Float ) : Float
7707:   Function uCosh( X : Float ) : Float
7708:   Function uTanh( X : Float ) : Float
7709:   Function uArcSinh( X : Float ) : Float
7710:   Function uArcCosh( X : Float ) : Float
7711:   Function ArcTanh( X : Float ) : Float
7712:   Procedure SinhCosh( X : Float; var SinhX, CoshX : Float )
7713: end;
7714:
7715: procedure SIRegister_urandom(CL: TPSPPascalCompiler);
7716: begin
7717:   type RNG_Type =
7718:     (RNG_MWC,           { Multiply-With-Carry }
7719:      RNG_MT,           { Mersenne Twister }
7720:      RNG_UVAG);        { Universal Virtual Array Generator }
7721:   Procedure SetRNG( RNG : RNG_Type )
7722:   Procedure InitGen( Seed : RNG_IntType )
7723:   Procedure SRand( Seed : RNG_IntType )
7724:   Function IRanGen : RNG_IntType
7725:   Function IRanGen31 : RNG_IntType
7726:   Function RanGen1 : Float
7727:   Function RanGen2 : Float
7728:   Function RanGen3 : Float
7729:   Function RanGen53 : Float
7730: end;
7731:
7732: // Optimization by Simulated Annealing
7733: procedure SIRegister_usimann(CL: TPSPPascalCompiler);
7734: begin
7735:   Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float )
7736:   Procedure SA_CreateLogFile( FileName : String )
7737:   Procedure SimAnn(Func: TFuncNVar; X,Xmin,Xmax: TVector; Lb, Ub : Integer; var F_min : Float );
7738: end;
7739:
7740: procedure SIRegister_uranuvag(CL: TPSPPascalCompiler);
7741: begin
7742:   Procedure InitUVAGbyString( KeyPhrase : string )
7743:   Procedure InitUVAG( Seed : RNG_IntType )
7744:   Function IRanUVAG : RNG_IntType
7745: end;
7746:
7747: procedure SIRegister_ugenalg(CL: TPSPPascalCompiler);
7748: begin
7749:   Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float )
7750:   Procedure GA_CreateLogFile( LogFileName : String )
7751:   Procedure GenAlg(Func: TFuncNVar; X,Xmin,Xmax : TVector; Lb, Ub : Integer; var F_min : Float );
7752: end;
7753:
7754: TVector', 'array of Float
7755: procedure SIRegister_uqsort(CL: TPSPPascalCompiler);
7756: begin
7757:   Procedure QSort( X : TVector; Lb, Ub : Integer )
7758:   Procedure DQSort( X : TVector; Lb, Ub : Integer )
7759: end;
7760:
7761: procedure SIRegister_uinterv(CL: TPSPPascalCompiler);
7762: begin
7763:   Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float )
7764:   Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float )
7765: end;
7766:
7767: procedure SIRegister_D2XXUnit(CL: TPSPPascalCompiler);
7768: begin
7769:   FT_Result', 'Integer

```

```

7770: //TDWordptr', '^DWord // will not work
7771: TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7772: d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7773: r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7774: ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B'
7775: yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7776: te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7777: ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7778: erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7779: Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7780: te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7781: yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7782: Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7783: nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7784: ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 : '
7785: Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsvCP : B'
7786: yte; end
7787: end;
7788:
7789:
7790: //***** PaintFX*****
7791: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7792: begin
7793: //with RegClass(CL, 'TComponent', 'TJvPaintFX') do
7794: with AddClassN(FindClass('TComponent'), 'TJvPaintFX') do begin
7795: Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7796: Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7797: Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7798: Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7799: Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7800: Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7801: Procedure Turn( Src, Dst : TBitmap)
7802: Procedure TurnRight( Src, Dst : TBitmap)
7803: Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7804: Procedure TexturizeFile( const Dst : TBitmap; Amount : Integer)
7805: Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7806: Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7807: Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7808: Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7809: Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7810: Procedure DrawMandelJulia(const Dst:TBitmap;x0,y0,x1,y1:single;Niter:Integer;Mandel:Bool)
7811: Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7812: Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7813: Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7814: Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7815: Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7816: Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7817: Procedure Emboss( var Bmp : TBitmap)
7818: Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7819: Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7820: Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7821: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7822: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7823: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7824: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7825: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7826: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7827: Procedure QuartoOpaque( Src, Dst : TBitmap)
7828: Procedure SemiOpaque( Src, Dst : TBitmap)
7829: Procedure ShadowDownLeft( const Dst : TBitmap)
7830: Procedure ShadowDownRight( const Dst : TBitmap)
7831: Procedure ShadowUpLeft( const Dst : TBitmap)
7832: Procedure ShadowUpRight( const Dst : TBitmap)
7833: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7834: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7835: Procedure FlipRight( const Dst : TBitmap)
7836: Procedure FlipDown( const Dst : TBitmap)
7837: Procedure Spotlight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7838: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7839: Procedure MakeSeamlessClip( var Dst : TBitmap; Sean : Integer)
7840: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7841: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7842: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7843: Procedure SmoothResize( var Src, Dst : TBitmap)
7844: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7845: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7846: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7847: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7848: Procedure GrayScale( const Dst : TBitmap)
7849: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7850: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7851: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7852: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7853: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7854: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7855: Procedure AntiAlias( const Dst : TBitmap)
7856: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7857: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7858: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)

```

```

7859: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7860: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7861: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7862: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7863: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7864: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7865: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7866: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7867: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7868: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7869: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7870: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7871: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7872: Procedure Invert( Src : TBitmap)
7873: Procedure MirrorRight( Src : TBitmap)
7874: Procedure MirrorDown( Src : TBitmap)
7875: end;
7876: end;
7877:
7878: (*-----*)
7879: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7880: begin
7881:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7882:   +'ye, lbrotate, lbtwist, lbrimble, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7883:   +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7884:   SIRegister_TJvPaintFX(CL);
7885:   Function SplineFilter( Value : Single) : Single
7886:   Function BellFilter( Value : Single) : Single
7887:   Function TriangleFilter( Value : Single) : Single
7888:   Function BoxFilter( Value : Single) : Single
7889:   Function HermiteFilter( Value : Single) : Single
7890:   Function Lanczos3Filter( Value : Single) : Single
7891:   Function MitchellFilter( Value : Single) : Single
7892: end;
7893:
7894:
7895: (*-----*)
7896: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7897: begin
7898:   'TeeMsg_DefaultFunctionName', 'String 'TeeFunction
7899:   TeeMsg_DefaultSeriesName', 'String 'Series
7900:   TeeMsg_DefaultToolName', 'String 'ChartTool
7901:   ChartComponentPalette', 'String 'TeeChart
7902:   TeeMaxLegendColumns', 'LongInt'( 2);
7903:   TeeDefaultLegendSymbolWidth', 'LongInt'( 20);
7904:   TeeTitleFootDistance, 'LongInt( 5);
7905:   SIRegister_TCustomChartWall(CL);
7906:   SIRegister_TChartWall(CL);
7907:   SIRegister_TChartLegendGradient(CL);
7908:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7909:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7910:   FindClass('TOBJECT'), 'LegendException
7911:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7912:   +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7913:   FindClass('TOBJECT'), 'TCustomChartLegend
7914:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7915:   TLegendSymbolPosition', '( spLeft, spRight )
7916:   TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect)';
7917:   TSymbolCalcHeight', 'Function : Integer
7918:   SIRegister_TLegendSymbol(CL);
7919:   SIRegister_TTeeCustomShapePosition(CL);
7920:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7921:   SIRegister_TLegendTitle(CL);
7922:   SIRegister_TLegendItem(CL);
7923:   SIRegister_TLegendItems(CL);
7924:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7925:   FindClass('TOBJECT'), 'TCustomChart
7926:   SIRegister_TCustomChartLegend(CL);
7927:   SIRegister_TChartLegend(CL);
7928:   SIRegister_TChartTitle(CL);
7929:   SIRegister_TChartFootTitle(CL);
7930:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7931:   +'eButton; Shift : TShiftState; X, Y : Integer)
7932:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7933:   +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7934:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7935:   +'TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7936:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7937:   +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7938:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7939:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7940:   TAxissavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7941:   +'tоМax : Boolean; Min : Double; Max : Double; end
7942:   TA11AxissavedScales', 'array of TAxissavedScales
7943:   SIRegister_TChartBackWall(CL);
7944:   SIRegister_TChartRightWall(CL);
7945:   SIRegister_TChartBottomWall(CL);
7946:   SIRegister_TChartLeftWall(CL);
7947:   SIRegister_TChartWalls(CL);

```

```

7948: TChartAllowScrollEvent', 'Procedure (Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7949: SIRегистer_TCustomChart(CL);
7950: SIRегистer_TChart(CL);
7951: SIRегистer_TTeeSeriesTypes(CL);
7952: SIRегистer_TTeeToolTypes(CL);
7953: SIRегистer_TTeeDragObject(CL);
7954: SIRегистер_TColorPalettes(CL);
7955: Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
AGalleryPage:PString;ANumGallerySeries:Integer);
7956: Procedure RegisterTeeSeries( ASeriesClass : TChartSeriesClass; ADescription : PString);
7957: Procedure RegisterTeeFunction(AFuncClass:TTeeFunctionClass;ADescription,
AGalleryPage:PString;ANumGallerySeries : Int;
7958: Procedure RegisterTeeBasicFunction(AFunctionClass: TTeeFunctionClass; ADescription : PString)
7959: Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7960: Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7961: Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7962: Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7963: Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7964: Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
AFunctionClass : TTeeFunctionClass) : TChartSeries
7965: Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7966: Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7967: Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7968: Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent ) : TTeeCustomTool
7969: Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7970: Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7971: Function GetNewSeriesName( AOwner : TComponent ) : TComponentName
7972: Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7973: Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7974: Function GetGallerySeriesName( ASeries : TChartSeries) : String
7975: Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
R:TRect;ReferenceChart:TCustomChart);
7976: SIRегистер_TChartTheme(CL);
7977: //TChartThemeClass', 'class of TChartTheme
7978: //TCanvasClass', 'class of TCanvas3D
7979: Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7980: Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7981: Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7982: Procedure ShowMessageUser( const S : String)
7983: Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7984: Function HasLabels( ASeries : TChartSeries) : Boolean
7985: Function HasColors( ASeries : TChartSeries) : Boolean
7986: Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7987: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7988: end;
7989:
7990:
7991: procedure SIRегистер_TeeProcs(CL: TPSPascalCompiler);
7992: begin
7993: //''TeeFormBorderStyle',' bsNone);
7994: SIRегистер_TMetafile(CL);
7995: 'TeeDefVerticalMargin','LongInt'( 4 );
7996: 'TeeDefHorizMargin','LongInt'( 3 );
7997: 'crTeeHand','LongInt'( TCursor ( 2020 ) );
7998: 'TeeMsg_TeeHand','String 'crTeeHand
7999: 'TeeNormalPrintDetail','LongInt'( 0 );
8000: 'TeeHighPrintDetail','LongInt'( - 100 );
8001: 'TeeDefault_PrintMargin','LongInt'( 15 );
8002: 'MaxDefaultColors','LongInt'( 19 );
8003: 'TeeTabDelimiter','Char #9';
8004: 'TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
8005: '+nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
8006: '+inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
8007: '+neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
8008: '+eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
8009: +'ourMonths, dtSixMonths, dtOneYear, dtNone )
8010: SIRегистер_TCustomPanelNoCaption(CL);
8011: FindClass('TOBJECT'),'TCustomTeePanel
8012: SIRегистер_TZoomPanning(CL);
8013: SIRегистер_TTeeEvent(CL);
8014: //SIRегистер_TTeeEventListeners(CL);
8015: TTeeMouseEventKind', '( meDown, meUp, meMove )
8016: SIRегистер_TTeeMouseEvent(CL);
8017: SIRегистер_TCustomTeePanel(CL);
8018: //TChartGradient', 'TTeeGradient
8019: //TChartGradientClass', 'class of TChartGradient
8020: TFanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
8021: SIRегистер_TTeeZoomPen(CL);
8022: SIRегистер_TTeeZoomBrush(CL);
8023: TTeeZoomDirection, '( tzdHorizontal, tzdVertical, tzdBoth )
8024: SIRегистер_TTeeZoom(CL);
8025: FindClass('TOBJECT'),'TCustomTeePanelExtended
8026: TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
8027: SIRегистер_TBackImage(CL);
8028: SIRегистер_TCustomTeePanelExtended(CL);
8029: //TChartBrushClass', 'class of TChartBrush
8030: SIRегистер_TTeeCustomShapeBrushPen(CL);
8031: TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )

```

```

8032: TTextFormat', '( ttfNormal, ttfHtml )
8033: SIRegister_TTeeCustomShape(CL);
8034: SIRegister_TTeeShape(CL);
8035: SIRegister_TTeeExportData(CL);
8036: Function TeeStr( const Num : Integer ) : String
8037: Function DateTimeDefaultFormat( const AStep : Double ) : String
8038: Function TEEDaysInMonth( Year, Month : Word ) : Word
8039: Function FindDateTimeStep( const StepValue : Double ) : TDateTimeStep
8040: Function NextDateTimeStep( const AStep : Double ) : Double
8041: Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
8042: Function PointInLine1( const P, FromPoint, ToPoint : TPoint ) : Boolean;
8043: Function PointInLine2( const P, FromPoint, ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
8044: Function PointInLine3( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
8045: Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
8046: Function PointInPolygon( const P : TPoint; const Poly : array of TPoint ) : Boolean
8047: Function PointInTriangle( const P, P0, P1, P2 : TPoint ) : Boolean;
8048: Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer ) : Boolean;
8049: Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer ) : Boolean
8050: Function PointInEllipse( const P : TPoint; const Rect : TRect ) : Boolean;
8051: Function PointInEllipSel( const P: TPoint;Left,Top,Right, Bottom : Integer ) : Boolean;
8052: Function DelphiToLocalFormat( const Format : String ) : String
8053: Function LocalToDelphiFormat( const Format : String ) : String
8054: Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8055: Function TeeRoundDate( const ADate : TDateTime; AStep : TDateTimeStep ) : TDateTime
8056: Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8057: TTeeSortCompare', 'Function ( a, b : Integer ) : Integer
8058: TTeeSortSwap', 'Procedure ( a, b : Integer )
8059: Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8060: Function TeeGetUniqueName( AOwner : TComponent; const AStartTime : String ) : string
8061: Function TeeExtractField( St : String; Index : Integer ) : String;
8062: Function TeeExtractField1( St : String; Index : Integer; const Separator : String ) : String;
8063: Function TeeNumFields( St : String ) : Integer;
8064: Function TeeNumFields1( const St, Separator : String ) : Integer;
8065: Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8066: Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8067: // TColorArray', 'array of TColor
8068: Function GetDefaultColor( const Index : Integer ) : TColor
8069: Procedure SetDefaultColorPalette;
8070: Procedure SetDefaultColorPalettes( const Palette : array of TColor);
8071: 'TeeCheckBoxSize','LongInt'( 11 );
8072: Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8073: Function TEEStrToFloatDef( const S : string; const Default : Extended ) : Extended
8074: Function TryStrToFloat( const S : String; var Value : Double ) : Boolean
8075: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double ) : Boolean
8076: Procedure TeeTranslateControl( AControl : TControl );
8077: Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl );
8078: Function ReplaceChar(const AString:String;const Search: Char; const Replace : Char) : String
8079: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8080: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean ) : TBitmap
8081: //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8082: //Function ScreenRatio( ACanvas : TCanvas3D ) : Double
8083: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean ) : Boolean
8084: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8085: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer ) : Integer
8086: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8087: Function TeeReadStringOption( const AKey, DefaultValue : String ) : String
8088: Procedure TeeSaveStringOption( const AKey, Value : String)
8089: Function TeeDefaultXMLEncoding : String
8090: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8091: TeeWindowHandle', 'Integer
8092: Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8093: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8094: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String ) : TSize
8095: end;
8096:
8097:
8098: using mXBDEUtils
8099: ****
8100: Procedure SetAlias( aAlias, aDirectory : String)
8101: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
8102: Function GetFileVersionNumber( const FileName : String ) : TVersionNo
8103: Procedure SetBDE( aPath, aNode, aValue : String)
8104: function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8105: Function GetSystemDirectory( lpBuffer : string; uSize : UINT ) : UINT
8106: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT ) : UINT
8107: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string ) : DWORD
8108: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string ) : DWORD
8109: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8110:
8111:
8112: procedure SIRegister_cDateTime(CL: TPPascalCompiler);
8113: begin
8114: AddClassN(FindClass('TOBJECT'),'EDateTime'
8115: Function DatePart( const D : TDateTime ) : Integer
8116: Function TimePart( const D : TDateTime ) : Double
8117: Function Century( const D : TDateTime ) : Word
8118: Function Year( const D : TDateTime ) : Word

```

```

8119: Function Month( const D : TDateTime) : Word
8120: Function Day( const D : TDateTime) : Word
8121: Function Hour( const D : TDateTime) : Word
8122: Function Minute( const D : TDateTime) : Word
8123: Function Second( const D : TDateTime) : Word
8124: Function Millisecond( const D : TDateTime) : Word
8125: ('OneDay','Extended').SetExtended( 1.0);
8126: ('OneHour','Extended').SetExtended( OneDay / 24);
8127: ('OneMinute','Extended').SetExtended( OneHour / 60);
8128: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8129: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8130: ('OneWeek','Extended').SetExtended( OneDay * 7);
8131: ('HoursPerDay','Extended').SetExtended( 24);
8132: ('MinutesPerHour','Extended').SetExtended( 60);
8133: ('SecondsPerMinute','Extended').SetExtended( 60);
8134: Procedure SetYear( var D : TDateTime; const Year : Word)
8135: Procedure SetMonth( var D : TDateTime; const Month : Word)
8136: Procedure SetDay( var D : TDateTime; const Day : Word)
8137: Procedure SetHour( var D : TDateTime; const Hour : Word)
8138: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8139: Procedure SetSecond( var D : TDateTime; const Second : Word)
8140: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8141: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8142: Function IsEqual( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8143: Function IsEqual( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8144: Function IsAM( const D : TDateTime) : Boolean
8145: Function IsPM( const D : TDateTime) : Boolean
8146: Function IsMidnight( const D : TDateTime) : Boolean
8147: Function IsNoon( const D : TDateTime) : Boolean
8148: Function IsSunday( const D : TDateTime) : Boolean
8149: Function IsMonday( const D : TDateTime) : Boolean
8150: Function IsTuesday( const D : TDateTime) : Boolean
8151: Function IsWednesday( const D : TDateTime) : Boolean
8152: Function IsThursday( const D : TDateTime) : Boolean
8153: Function IsFriday( const D : TDateTime) : Boolean
8154: Function IsSaturday( const D : TDateTime) : Boolean
8155: Function IsWeekend( const D : TDateTime) : Boolean
8156: Function Noon( const D : TDateTime) : TDateTime
8157: Function Midnight( const D : TDateTime) : TDateTime
8158: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8159: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8160: Function NextWorkday( const D : TDateTime) : TDateTime
8161: Function PreviousWorkday( const D : TDateTime) : TDateTime
8162: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8163: Function LastDayOfYear( const D : TDateTime) : TDateTime
8164: Function EasterSunday( const Year : Word) : TDateTime
8165: Function GoodFriday( const Year : Word) : TDateTime
8166: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8167: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8168: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8169: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8170: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8171: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8172: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8173: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8174: Function DayOffYear( const Ye, Mo, Da : Word) : Integer
8175: Function DayOfYear( const D : TDateTime) : Integer
8176: Function DaysInMonth( const Ye, Mo : Word) : Integer
8177: Function DaysInMonth( const D : TDateTime) : Integer
8178: Function DaysInYear( const Ye : Word) : Integer
8179: Function DaysInYearDate( const D : TDateTime) : Integer
8180: Function WeekNumber( const D : TDateTime) : Integer
8181: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8182: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8183: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8184: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8185: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8186: Function DiffHours( const D1, D2 : TDateTime) : Integer
8187: Function DiffDays( const D1, D2 : TDateTime) : Integer
8188: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8189: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8190: Function DiffYears( const D1, D2 : TDateTime) : Integer
8191: Function GMTBias : Integer
8192: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8193: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8194: Function NowAsGMTTime : TDateTime
8195: Function DatetimeToISO8601String( const D : TDateTime) : AnsiString
8196: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8197: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8198: Function DateTimeToANSI( const D : TDateTime) : Integer
8199: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8200: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8201: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8202: Function ISOIntegerToDate( const ISOInteger : Integer) : TDateTime
8203: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8204: Function DateTimeAsElapsedTime( const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8205: Function UnixTimeToDate( const UnixTime : LongWord) : TDateTime
8206: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8207: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString

```

```

8208: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer ) : UnicodeString
8209: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer ) : AnsiString
8210: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer ) : UnicodeString
8211: Function EnglishShortMonthStrA( const Month : Integer ) : AnsiString
8212: Function EnglishShortMonthStrU( const Month : Integer ) : UnicodeString
8213: Function EnglishLongMonthStrA( const Month : Integer ) : AnsiString
8214: Function EnglishLongMonthStrU( const Month : Integer ) : UnicodeString
8215: Function EnglishShortDayOfWeekA( const S : AnsiString ) : Integer
8216: Function EnglishShortDayOfWeekU( const S : UnicodeString ) : Integer
8217: Function EnglishLongDayOfWeekA( const S : AnsiString ) : Integer
8218: Function EnglishLongDayOfWeekU( const S : UnicodeString ) : Integer
8219: Function EnglishShortMonthA( const S : AnsiString ) : Integer
8220: Function EnglishShortMonthU( const S : UnicodeString ) : Integer
8221: Function EnglishLongMonthA( const S : AnsiString ) : Integer
8222: Function EnglishLongMonthU( const S : UnicodeString ) : Integer
8223: Function RFC850DayOfWeekA( const S : AnsiString ) : Integer
8224: Function RFC850DayOfWeekU( const S : UnicodeString ) : Integer
8225: Function RFC1123DayOfWeekA( const S : AnsiString ) : Integer
8226: Function RFC1123DayOfWeekU( const S : UnicodeString ) : Integer
8227: Function RFCMonthA( const S : AnsiString ) : Word
8228: Function RFCMonthU( const S : UnicodeString ) : Word
8229: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean ) : AnsiString
8230: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean ) : UnicodeString
8231: Function GMTDateTimeToRFC1123DateTimeA( const D : TDateTime; const IncludeDayOfWeek:Bool ):AnsiString;
8232: Function GMTDateTimeToRFC1123DateTimeU( const D:TDateTime;const IncludeDayOfWeek:Bool ):UnicodeString;
8233: Function DateTimeToRFCDateTimeA( const D : TDateTime ) : AnsiString
8234: Function DateTimeToRFCDateTimeU( const D : TDateTime ) : UnicodeString
8235: Function NowAsRFCDateTimeA : AnsiString
8236: Function NowAsRFCDateTimeU : UnicodeString
8237: Function RFCDateTimeToGMTDateTime( const S : AnsiString ) : TDateTime
8238: Function RFCDateTimeToDateTIme( const S : AnsiString ) : TDateTime
8239: Function RFCTimeZoneToGMTBias( const Zone : AnsiString ) : Integer
8240: Function TimePeriodStr( const D : TDateTime ) : AnsiString
8241: Procedure SelfTest
8242: end;
8243: //*****CFileUtils
8244: Function PathHasDriveLetterA( const Path : AnsiString ) : Boolean
8245: Function PathHasDriveLetter( const Path : String ) : Boolean
8246: Function PathIsDriveLetterA( const Path : AnsiString ) : Boolean
8247: Function PathIsDriveLetter( const Path : String ) : Boolean
8248: Function PathIsDriveRootA( const Path : AnsiString ) : Boolean
8249: Function PathIsDriveRoot( const Path : String ) : Boolean
8250: Function PathIsRootA( const Path : AnsiString ) : Boolean
8251: Function PathIsRoot( const Path : String ) : Boolean
8252: Function PathIsUNCPathA( const Path : AnsiString ) : Boolean
8253: Function PathIsUNCPath( const Path : String ) : Boolean
8254: Function PathIsAbsoluteA( const Path : AnsiString ) : Boolean
8255: Function PathIsAbsolute( const Path : String ) : Boolean
8256: Function PathIsDirectoryA( const Path : AnsiString ) : Boolean
8257: Function PathIsDirectory( const Path : String ) : Boolean
8258: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8259: Function PathInclSuffix( const Path : String; const PathSep : Char ) : String
8260: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8261: Function PathExclSuffix( const Path : String; const PathSep : Char ) : String
8262: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char )
8263: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char )
8264: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char )
8265: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char )
8266: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8267: Function PathCanonical( const Path : String; const PathSep : Char ) : String
8268: Function PathExpandA(const Path:AnsiString;const basePath:AnsiString;const PathSep:Char):AnsiString
8269: Function PathExpand( const Path : String; const basePath : String; const PathSep : Char ) : String
8270: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8271: Function PathLeftElement( const Path : String; const PathSep : Char ) : String
8272: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8273: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8274: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char);
8275: Procedure DecodeFilePath(const FilePath:String; var Path,FileName:String;const PathSep:Char)
8276: Function FileNameValidA( const FileName : AnsiString ) : AnsiString
8277: Function FileNameValid( const FileName : String ) : String
8278: Function FilePathA(const FileName,Path:AnsiString;const basePath:AnsiStr;const PathSep:Char):AnsiString;
8279: Function FilePath(const FileName, Path : String;const basePath : String;const PathSep : Char) : String
8280: Function DirectoryExpandA(const Path:AnsiString;const basePath:AnsiString;const PathSep:Char):AnsiString
8281: Function DirectoryExpand(const Path : String; const basePath : String; const PathSep : Char ) : String
8282: Function UnixPathToWinPath( const Path : AnsiString ) : AnsiString
8283: Function WinPathToUnixPath( const Path : AnsiString ) : AnsiString
8284: Procedure CCopyFile( const FileName, DestName : String)
8285: Procedure CMoveFile( const FileName, DestName : String)
8286: Function CDeleteFiles( const FileMode : String) : Boolean
8287: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8288: Procedure FileCloseEx( const FileHandle : TFileHandle)
8289: Function FileExistsA( const FileName : AnsiString ) : Boolean
8290: Function CFileExists( const FileName : String) : Boolean
8291: Function CFileGetSize( const FileName : String) : Int64
8292: Function FileGetDateTime( const FileName : String ) : TDateTime
8293: Function FileGetDateTime2( const FileName : String ) : TDateTime
8294: Function FileIsReadOnly( const FileName : String ) : Boolean
8295: Procedure FileDeleteEx( const FileName : String)
8296: Procedure FileRenameEx( const OldFileName, NewFileName : String)

```

```

8297: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
8298:   : TFileCreationMode; const FileOpenWait : PFileOpenWait ) : AnsiString
8299: Function DirectoryEntryExists( const Name : String ) : Boolean
8300: Function DirectoryEntrySize( const Name : String ) : Int64
8301: Function CDirectoryExists( const DirectoryName : String ) : Boolean
8302: Function DirectoryGetDateTime( const DirectoryName : String ) : TDateTime
8303: Procedure CDirectoryCreate( const DirectoryName : String )
8304: Function GetFirstFileNameMatching( const FileMask : String ) : String
8305: Function DirEntryGetAttr( const FileName : AnsiString ) : Integer
8306: Function DirEntryIsDirectory( const FileName : AnsiString ) : Boolean
8307: Function FileHasAttr( const FileName : String; const Attr : Word ) : Boolean
8308:   AddTypesS('TLogicalDriveType','( DriveRemovable, DriveFixed, DriveRemote,
8309:     +DriveCDRom, DriveRamDisk, DriveTypeUnknown )')
8310: Function DriveIsValid( const Drive : Char ) : Boolean
8311: Function DriveGetType( const Path : AnsiString ) : TLogicalDriveType
8312: Function DriveFreeSpace( const Path : AnsiString ) : Int64
8313: procedure SIRegister_cTimers(CL: TPSPPascalCompiler);
8314: begin
8315:   AddClassN(FindClass('TOBJECT'), 'ETimers'
8316:     Const('TickFrequency', 'LongInt'( 1000 ); Function GetTick : LongWord
8317:   Function TickDelta( const D1, D2 : LongWord ) : Integer
8318:   Function TickDeltaW( const D1, D2 : LongWord ) : LongWord
8319:     AddTypesS('THPTimer', 'Int64'
8320:   Procedure StartTimer( var Timer : THPTimer )
8321:   Procedure StopTimer( var Timer : THPTimer )
8322:   Procedure ResumeTimer( var StoppedTimer : THPTimer )
8323:   Procedure InitStoppedTimer( var Timer : THPTimer )
8324:   Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer )
8325:   Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean ) : Integer
8326:   Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean ) : Int64
8327:   Procedure WaitMicroseconds( const MicroSeconds : Integer )
8328:   Function GetHighPrecisionFrequency : Int64
8329:   Function GetHighPrecisionTimerOverhead : Int64
8330:   Procedure AdjustTimerForOverhead(var StoppedTimer: THPTimer; const Overhead : Int64)
8331:   Procedure SelfTestCTimer
8332: end;
8333:
8334: procedure SIRegister_cRandom(CL: TPSPPascalCompiler);
8335: begin
8336:   Function RandomSeed : LongWord
8337:   Procedure AddEntropy( const Value : LongWord )
8338:   Function RandomUniform : LongWord;
8339:   Function RandomUniforml( const N : Integer ) : Integer;
8340:   Function RandomBoolean : Boolean
8341:   Function RandomByte : Byte
8342:   Function RandomByteNonZero : Byte
8343:   Function RandomWord : Word
8344:   Function RandomInt64 : Int64;
8345:   Function RandomInt641( const N : Int64 ) : Int64;
8346:   Function RandomHex( const Digits : Integer ) : String
8347:   Function RandomFloat : Extended
8348:   Function RandomAlphaStr( const Length : Integer ) : AnsiString
8349:   Function RandomPseudoWord( const Length : Integer ) : AnsiString
8350:   Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8351:   Function mwcRandomLongWord : LongWord
8352:   Function urnRandomLongWord : LongWord
8353:   Function moaRandomFloat : Extended
8354:   Function mwcRandomFloat : Extended
8355:   Function RandomNormalF : Extended
8356:   Procedure SelfTestCRandom
8357: end;
8358:
8359: procedure SIRegister_SynEditMiscProcs(CL: TPSPPascalCompiler);
8360: begin
8361:   // PIntArray', '^TIntArray // will not work
8362:   Addtypes('TConvertTabsProc', 'function(const Line:AnsiString; TabWidth: integer):AnsiString
8363:   TConvertTabsProcEx, function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8364:   Function synMax( x, y : integer ) : integer
8365:   Function synMin( x, y : integer ) : integer
8366:   Function synMinMax( x, mi, ma : integer ) : integer
8367:   Procedure synSwapInt( var l, r : integer )
8368:   Function synMaxPoint( const P1, P2 : TPoint ) : TPoint
8369:   Function synMinPoint( const P1, P2 : TPoint ) : TPoint
8370:   //Function synGetIntArray( Count : Cardinal; initialValue : integer ) : PIntArray
8371:   Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect )
8372:   Function synGetBestConvertTabsProc( TabWidth : integer ) : TConvertTabsProc
8373:   Function synConvertTabs( const Line : AnsiString; TabWidth : integer ) : AnsiString
8374:   Function synGetBestConvertTabsProcEx( TabWidth : integer ) : TConvertTabsProcEx
8375:   Function synConvertTabsEx( const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8376:   Function synGetExpandedLength( const aStr : string; aTabWidth : integer ) : integer
8377:   Function synCharIndex2Caretpos( Index, TabWidth : integer; const Line : string ) : integer
8378:   Function synCaretpos2CharIndex( Position, TabWidth:int;const Line:string;var InsideTabChar:boolean ):int;
8379:   Function synStrScanForCharInSet( const Line:string;Start:integer;AChars:TSynIdentChars):int;
8380:   Function synStrRScanForCharInSet( const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8381:   TStringType', '( stNone, stHalfNumAlpha, stWideNumAlpha, stHalfSymbol, stHalfKat'
8382:     +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8383:   ('C3_NONSPACING','LongInt'( 1 );
8384:   'C3_DIACRITIC','LongInt'( 2 );

```

```

8385:   'C3_VOWELMARK','LongInt'( 4);
8386:   ('C3_SYMBOL','LongInt'( 8);
8387:   ('C3_KATAKANA','LongWord( $0010);
8388:   ('C3_HIRAGANA','LongWord( $0020);
8389:   ('C3_HALFWIDTH','LongWord( $0040);
8390:   ('C3_FULLWIDTH','LongWord( $0080);
8391:   ('C3_IDEOGRAPH','LongWord( $0100);
8392:   ('C3_KASHIDA','LongWord( $0200);
8393:   ('C3_LEXICAL','LongWord( $0400);
8394:   ('C3_ALPHA','LongWord( $8000);
8395:   ('C3_NOTAPPLICABLE','LongInt'( 0);
8396:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8397:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8398:   Function synIsStringType( Value : Word) : TStringType
8399:   Function synGetEOL( Line : PChar) : PChar
8400:   Function synEncodeString( s : string) : string
8401:   Function synDecodeString( s : string) : string
8402:   Procedure synFreeAndNil( var Obj: TObject)
8403:   Procedure synAssert( Expr : Boolean)
8404:   Function synLastDelimiter( const Delimiters, S : string) : Integer
8405:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8406:   TReplaceFlags', 'set of TReplaceFlag )
8407:   Function synStringReplace(const S, OldPattern,NewPattern:string;Flags:TReplaceFlags) : string
8408:   Function synGetRValue( RGBValue : TColor) : byte
8409:   Function synGetGValue( RGBValue : TColor) : byte
8410:   Function synGetBValue( RGBValue : TColor) : byte
8411:   Function synRGB( r, g, b : Byte) : Cardinal
8412: // THighlighterAttriProc', 'Function( Highlighter : TSynCustomHigh'
8413: // +'lighter; Attr:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8414: //Function synEnumHighlighterAttrs( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
8415: HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer ) : Boolean
8416:   Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8417:   Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8418:   end;
8419:   Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8420:   Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8421:   Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8422:
8423: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8424: begin
8425:   Function STimeZoneBias : integer
8426:   Function TimeZone : string
8427:   Function Rfc222DateTIme( t : TDateTIme) : string
8428:   Function CDateTIme( t : TDateTIme) : string
8429:   Function SimpleDateTIme( t : TDateTIme) : string
8430:   Function AnsiCDateTIme( t : TDateTIme) : string
8431:   Function GetMonthNumber( Value : String) : integer
8432:   Function GettimeFromStr( Value : string) : TDateTIme
8433:   Function GetDateMDYFromStr( Value : string) : TDateTIme
8434:   Function DecodeRFCDateTIme( Value : string) : TDateTIme
8435:   Function GetUTTIme : TDateTIme
8436:   Function SetUTTIme( Newdt : TDateTIme) : Boolean
8437:   Function SGetTick : LongWord
8438:   Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8439:   Function CodeInt( Value : Word) : Ansistring
8440:   Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8441:   Function CodeLongInt( Value : LongInt) : Ansistring
8442:   Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8443:   Function DumpStr( const Buffer : Ansistring) : string
8444:   Function DumpExStr( const Buffer : Ansistring) : string
8445:   Procedure Dump( const Buffer : Ansistring; DumpFile : string)
8446:   Procedure DumpEx( const Buffer : Ansistring; DumpFile : string)
8447:   Function TrimSPLeft( const S : string) : string
8448:   Function TrimSPRight( const S : string) : string
8449:   Function TrimSP( const S : string) : string
8450:   Function SeparateLeft( const Value, Delimiter : string) : string
8451:   Function SeparateRight( const Value, Delimiter : string) : string
8452:   Function SGetParameter( const Value, Parameter : string) : string
8453:   Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8454:   Procedure ParseParameters( Value : string; const Parameters : TStrings)
8455:   Function IndexByBegin( Value : string; const List : TStrings) : integer
8456:   Function GetEmailAddr( const Value : string) : string
8457:   Function GetEmailDesc( Value : string) : string
8458:   Function CStrToHex( const Value : Ansistring) : string
8459:   Function CIntToBin( Value : Integer; Digits : Byte) : string
8460:   Function CBinToInt( const Value : string) : Integer
8461:   Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8462:   Function CReplaceString( Value, Search, Replace : Ansistring) : Ansistring
8463:   Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8464:   Function CRPos( const Sub, Value : String) : Integer
8465:   Function FetchBin( var Value : string; const Delimiter : string) : string
8466:   Function CFetch( var Value : string; const Delimiter : string) : string
8467:   Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8468:   Function IsBinaryString( const Value : Ansistring) : Boolean
8469:   Function PosCRLF( const Value : Ansistring; var Terminator : Ansistring) : integer
8470:   Procedure StringsTrim( const value : TStrings)
8471:   Function PosFrom( const SubStr, Value : String; From : integer) : integer

```

```

8472: Function IncPoint( const p : __pointer; Value : integer) : __pointer
8473: Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8474: Function CCountOfChar( const Value : string; aChr : char) : integer
8475: Function UnquoteStr( const Value : string; Quote : Char) : string
8476: Function QuoteStr( const Value : string; Quote : Char) : string
8477: Procedure HeadersToList( const Value : TStrings)
8478: Procedure ListToHeaders( const Value : TStrings)
8479: Function SwapBytes( Value : integer) : integer
8480: Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8481: Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8482: Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8483: Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar) : AnsiString
8484: Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8485: Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8486: end;
8487:
8488: procedure SIRegister_StCRC(CL: TPSPPascalCompiler);
8489: begin
8490:   ('CrcBufSize','LongInt'( 2048);
8491:   Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8492:   Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8493:   Function Adler32OfFile( FileName : AnsiString) : LongInt
8494:   Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8495:   Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8496:   Function Crc16OfFile( FileName : AnsiString) : Cardinal
8497:   Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8498:   Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8499:   Function Crc32OfFile( FileName : AnsiString) : LongInt
8500:   Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8501:   Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8502:   Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8503:   Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8504:   Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8505:   Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8506: end;
8507:
8508: procedure SIRegister_ComObj(cl: TPSPPascalCompiler);
8509: begin
8510:   function CreateOleObject(const ClassName: String): IDispatch;
8511:   function GetActiveOleObject(const ClassName: String): IDispatch;
8512:   function ProgIDToClassID(const ProgID: string): TGUID;
8513:   function ClassIDToProgID(const ClassID: TGUID): string;
8514:   function CreateClassID: string;
8515:   function CreateGUIDString: string;
8516:   function CreateGUIDID: string;
8517:   procedure OleError(ErrorCode: longint)
8518:   procedure OleCheck(Result: HResult);
8519: end;
8520:
8521: Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8522: Function xGetActiveOleObject( const ClassName : string) : Variant
8523: //Function DllGetClassObject( const CLSID : TGUID; const IID : TIID; var Obj) : HResult
8524: Function DllCanUnloadNow : HResult
8525: Function DllRegisterServer : HResult
8526: Function DllUnregisterServer : HResult
8527: Function VarFromInterface( Unknown : IUnknown) : Variant
8528: Function VarToInterface( const V : Variant) : IDispatch
8529: Function VarToAutoObject( const V : Variant) : TAtoObject
8530: //Procedure
8531: DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8532: //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8533: Procedure OleError( ErrorCode : HResult)
8534: Procedure OleCheck( Result : HResult)
8535: Function StringToClassID( const S : string) : TGUID
8536: Function ClassIDToString( const ClassID : TGUID) : string
8537: Function xProgIDToClassID( const ProgID : string) : TGUID
8538: Function xClassIDToProgID( const ClassID : TGUID) : string
8539: Function xWideCompareStr( const S1, S2 : WideString) : Integer
8540: Function xGUIDToString( const ClassID : TGUID) : string
8541: Function xStringToGUID( const S : string) : TGUID
8542: Function xGetModuleName( Module : HMODULE) : string
8543: Function xAcquireExceptionObject : TObject
8544: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8545: Function xUtf8Encode( const WS : WideString) : UTF8String
8546: Function xUtf8Decode( const S : UTF8String) : WideString
8547: Function xExcludeTrailingPathDelimiter( const S : string) : string
8548: Function xIncludeTrailingPathDelimiter( const S : string) : string
8549: Function XRTLHandleCOMException : HResult
8550: Procedure XRTLCheckArgument( Flag : Boolean)
8551: //Procedure XRTLCheckOutArgument( out Arg)
8552: Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var Connection:Longint);
8553: Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8554: Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TGUID;Flags:DWORD;var RegisterCookie:Int):HResult
8555: Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8556: //Function XRTLGetActiveObject( ClassID : TGUID; RIID : TIID; out Obj) : HResult
8557: Procedure XRTLEnumActiveObjects( Strings : TStrings)

```

```

8558: function XRTLDefaultCategoryManager: IUnknown;
8559: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8560: // ICatRegister helper functions
8561: function XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8562:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8563:                                         const CategoryManager: IUnknown = nil): HResult;
8564: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8565:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8566:                                         const CategoryManager: IUnknown = nil): HResult;
8567: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8568:                                         const CategoryManager: IUnknown = nil): HResult;
8569: function XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8570:                                         const CategoryManager: IUnknown = nil): HResult;
8571: // ICatInformation helper functions
8572: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8573:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8574:                                         const CategoryManager: IUnknown = nil): HResult;
8575: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TGUID = LOCALE_USER_DEFAULT;
8576:                                         const CategoryManager: IUnknown = nil): HResult;
8577: function XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8578:                                         const CategoryManager: IUnknown = nil): HResult;
8579: function XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8580:                                         const CategoryManager: IUnknown = nil): HResult;
8581: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8582:                         const ADelete: Boolean = True): WideString;
8583: function XRTLPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8584: Function XRTLGetVariantAsString( const Value : Variant ) : string
8585: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone ) : TDateTime
8586: Function XRTLGetTimeZones : TXRTLTimeZones
8587: Function XFileTimeToDate( FileTime : TFileTime ) : TDateTime
8588: Function DateToFileTime( Date : TDateTime ) : TFileTime
8589: Function GMTNow : TDateTime
8590: Function GMTToLocalTime( GMT : TDateTime ) : TDateTime
8591: Function LocalTimeToGMT( LocalTime : TDateTime ) : TDateTime
8592: Procedure XRTLNotImplemented
8593: Procedure XRTLRaiseError( E : Exception )
8594: Procedure XRTLRaise( E : Exception );
8595: Procedure XRaise( E : Exception );
8596: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8597:
8598:
8599: procedure SIRegister_xrtl_util_Value(CL: TPPascalCompiler);
8600: begin
8601:   SIRegister_IXRTLValue(CL);
8602:   SIRegister_TXRTLValue(CL);
8603:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8604:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8605: Function XRTLValue( const AValue : Cardinal ) : IXRTLValue;
8606: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal ) : Cardinal;
8607: Function XRTLGetAsCardinal( const IValue : IXRTLValue ) : Cardinal;
8608: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal ) : Cardinal;
8609: Function XRTLValue1( const AValue : Integer ) : IXRTLValue;
8610: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer ) : Integer;
8611: Function XRTLGetAsInteger( const IValue : IXRTLValue ) : Integer;
8612: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer ) : Integer;
8613: Function XRTLValue2( const AValue : Int64 ) : IXRTLValue;
8614: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64 ) : Int64;
8615: Function XRTLGetAsInt64( const IValue : IXRTLValue ) : Int64;
8616: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64 ) : Int64;
8617: Function XRTLValue3( const AValue : Single ) : IXRTLValue;
8618: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single ) : Single;
8619: Function XRTLGetAsSingle( const IValue : IXRTLValue ) : Single;
8620: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single ) : Single;
8621: Function XRTLValue4( const AValue : Double ) : IXRTLValue;
8622: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double ) : Double;
8623: Function XRTLGetAsDouble( const IValue : IXRTLValue ) : Double;
8624: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double ) : Double;
8625: Function XRTLValue5( const AValue : Extended ) : IXRTLValue;
8626: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended ) : Extended;
8627: Function XRTLGetAsExtended( const IValue : IXRTLValue ) : Extended;
8628: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended ) : Extended;
8629: Function XRTLValue6( const AValue : IInterface ) : IXRTLValue;
8630: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface ) : IInterface;
8631: Function XRTLGetAsInterface( const IValue : IXRTLValue ) : IInterface;
8632: //Function XRTLGetAsInterface( const IValue : IXRTLValue; out Obj ) : IInterface;
8633: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface ) : IInterface;
8634: Function XRTLValue7( const AValue : WideString ) : IXRTLValue;
8635: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString ) : WideString;
8636: Function XRTLGetAsWideString( const IValue : IXRTLValue ) : WideString;
8637: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString ) : WideString;
8638: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean ) : IXRTLValue;
8639: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject ) : TObject;
8640: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership: Boolean ): TObject;
8641: Function XRTLGetAsObjectDef( const IValue:IXRTLValue;const DefValue:TObject;const
8642:                               ADetachOwnership:Boolean ):TObject;
8643: //Function XRTLValue9( const AValue : _Pointer ) : IXRTLValue;
8643: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : _Pointer ) : _Pointer;
8644: //Function XRTLGetAsPointer( const IValue : IXRTLValue ) : _Pointer
8645: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : _Pointer ) : _Pointer

```

```

8646: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8647: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8648: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8649: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8650: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8651: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8652: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8653: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8654: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8655: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8656: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8657: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8658: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8659: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8660: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8661: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8662: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8663: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8664: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8665: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8666: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8667: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8668: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8669: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8670: end;
8671:
8672: //*****unit uPSI_GR32;*****
8673:
8674: Function Color32( WinColor : TColor) : TColor32;
8675: Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8676: Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8677: Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8678: Function WinColor( Color32 : TColor32) : TColor
8679: Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8680: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8681: Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8682: Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8683: Function RedComponent( Color32 : TColor32) : Integer
8684: Function GreenComponent( Color32 : TColor32) : Integer
8685: Function BlueComponent( Color32 : TColor32) : Integer
8686: Function AlphaComponent( Color32 : TColor32) : Integer
8687: Function Intensity( Color32 : TColor32) : Integer
8688: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8689: Function HSLtoRGB( H, S, L : Single) : TColor32;
8690: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8691: Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8692: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8693: Function WinPalette( const P : TPalette32) : HPALETTE
8694: Function FloatPoint( X, Y : Single) : TFloatPoint;
8695: Function FloatPoint1( const P : TPoint) : TFfloatPoint;
8696: Function FloatPoint2( const FXP : TFixedPoint) : TFfloatPoint;
8697: Function FixedPoint( X, Y : Integer) : TFixedPoint;
8698: Function FixedPoint1( X, Y : Single) : TFixedPoint;
8699: Function FixedPoint2( const P : TPoint) : TFixedPoint;
8700: Function FixedPoint3( const FP : TFfloatPoint) : TFixedPoint;
8701: AddTypes('TRectRounding', '( rrClosest, rrOutside, rrInside )'
8702: Function MakeRect( const L, T, R, B : Integer) : TRect;
8703: Function MakeRect1( const FR : TFfloatRect; Rounding : TRectRounding) : TRect;
8704: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8705: Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8706: Function FixedRect1( const ARect : TRect) : TRect;
8707: Function FixedRect2( const FR : TFfloatRect) : TRect;
8708: Function GFloatRect( const L, T, R, B : TFfloat) : TFfloatRect;
8709: Function FloatRect1( const ARect : TRect) : TFfloatRect;
8710: Function FloatRect2( const FXR : TRect) : TFfloatRect;
8711: Function GIIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8712: Function IntersectRect1( out Dst : TFfloatRect; const FR1, FR2 : TFfloatRect) : Boolean;
8713: Function GUUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8714: Function UnionRect1( out Rect : TFfloatRect; const R1, R2 : TFfloatRect) : Boolean;
8715: Function GEEqualRect( const R1, R2 : TRect) : Boolean;
8716: Function EqualRect1( const R1, R2 : TFfloatRect) : Boolean;
8717: Procedure GIInflateRect( var R : TRect; Dx, Dy : Integer);
8718: Procedure InflateRect1( var FR : TFfloatRect; Dx, Dy : TFfloat);
8719: Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8720: Procedure OffsetRect1( var FR : TFfloatRect; Dx, Dy : TFfloat);
8721: Function IsRectEmpty( const R : TRect) : Boolean;
8722: Function IsRectEmpty1( const FR : TFfloatRect) : Boolean;
8723: Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8724: Function PtInRect1( const R : TFfloatRect; const P : TPoint) : Boolean;
8725: Function PtInRect2( const R : TRect; const P : TFfloatPoint) : Boolean;
8726: Function PtInRect3( const R : TFfloatRect; const P : TFfloatPoint) : Boolean;
8727: Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8728: Function EqualRectSize1( const R1, R2 : TFfloatRect) : Boolean;
8729: Function MessageBeep( uType : UINT) : BOOL
8730: Function ShowCursor( bShow : BOOL) : Integer
8731: Function SetCursorPos( X, Y : Integer) : BOOL
8732: Function SetCursor( hCursor : HICON) : HCURSOR
8733: Function GetCursorPos( var lpPoint : TPoint) : BOOL
8734: //Function ClipCursor( lpRect : PRect) : BOOL

```

```

8735: Function GetClipCursor( var lpRect : TRect ) : BOOL
8736: Function GetCursor : HCURSOR
8737: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer ) : BOOL
8738: Function GetCaretBlinkTime : UINT
8739: Function SetCaretBlinkTime( uSeconds : UINT ) : BOOL
8740: Function DestroyCaret : BOOL
8741: Function HideCaret( hWnd : HWND ) : BOOL
8742: Function ShowCaret( hWnd : HWND ) : BOOL
8743: Function SetCaretPos( X, Y : Integer ) : BOOL
8744: Function GetCaretPos( var lpPoint : TPoint ) : BOOL
8745: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8746: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8747: Function MapWindowPoints( hWndFrom, hWndTo : HWND; var lpPoints, cPoints : UINT ) : Integer
8748: Function WindowFromPoint( Point : TPoint ) : HWND
8749: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint ) : HWND
8750:
8751:
8752: procedure SIRegister_GR32_Math(CL: TPSPPascalCompiler);
8753: begin
8754:   Function FixedFloor( A : TFixed ) : Integer
8755:   Function FixedCeil( A : TFixed ) : Integer
8756:   Function FixedMul( A, B : TFixed ) : TFixed
8757:   Function FixedDiv( A, B : TFixed ) : TFixed
8758:   Function OneOver( Value : TFixed ) : TFixed
8759:   Function FixedRound( A : TFixed ) : Integer
8760:   Function FixedSqr( Value : TFixed ) : TFixed
8761:   Function FixedSqrtLP( Value : TFixed ) : TFixed
8762:   Function FixedSqrtHP( Value : TFixed ) : TFixed
8763:   Function FixedCombine( W, X, Y : TFixed ) : TFixed
8764:   Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat );
8765:   Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single );
8766:   Function GRHypot( const X, Y : TFloat ) : TFloat;
8767:   Function Hypot1( const X, Y : Integer ) : Integer;
8768:   Function FastSqrt( const Value : TFloat ) : TFloat
8769:   Function FastSqrtBab1( const Value : TFloat ) : TFloat
8770:   Function FastSqrtBab2( const Value : TFloat ) : TFloat
8771:   Function FastInvSqrt( const Value : Single ) : Single;
8772:   Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer
8773:   Function GRIsPowerOf2( Value : Integer ) : Boolean
8774:   Function PrevPowerOf2( Value : Integer ) : Integer
8775:   Function NextPowerOf2( Value : Integer ) : Integer
8776:   Function Average( A, B : Integer ) : Integer
8777:   Function GRSign( Value : Integer ) : Integer
8778:   Function FloatMod( x, y : Double ) : Double
8779: end;
8780:
8781: procedure SIRegister_GR32_LowLevel(CL: TPSPPascalCompiler);
8782: begin
8783:   Function Clamp( const Value : Integer ) : Integer;
8784:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword )
8785:   Function StackAlloc( Size : Integer ) : Pointer
8786:   Procedure StackFree( P : Pointer )
8787:   Procedure Swap( var A, B : Pointer );
8788:   Procedure Swap1( var A, B : Integer );
8789:   Procedure Swap2( var A, B : TFixed );
8790:   Procedure Swap3( var A, B : TColor32 );
8791:   Procedure TestSwap( var A, B : Integer );
8792:   Procedure TestSwap1( var A, B : TFixed );
8793:   Function TestClip( var A, B : Integer; const Size : Integer ) : Boolean;
8794:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer ) : Boolean;
8795:   Function GRConstrain( const Value, Lo, Hi : Integer ) : Integer;
8796:   Function Constrain1( const Value, Lo, Hi : Single ) : Single;
8797:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer ) : Integer
8798:   Function GRMin( const A, B, C : Integer ) : Integer;
8799:   Function GRMax( const A, B, C : Integer ) : Integer;
8800:   Function Clamp( Value, Max : Integer ) : Integer;
8801:   Function Clamp1( Value, Min, Max : Integer ) : Integer;
8802:   Function Wrap( Value, Max : Integer ) : Integer;
8803:   Function Wrap1( Value, Min, Max : Integer ) : Integer;
8804:   Function Wrap3( Value, Max : Single ) : Single;;
8805:   Function WrapPow2( Value, Max : Integer ) : Integer;
8806:   Function WrapPow21( Value, Min, Max : Integer ) : Integer;
8807:   Function Mirror( Value, Max : Integer ) : Integer;
8808:   Function Mirror1( Value, Min, Max : Integer ) : Integer;
8809:   Function MirrorPow2( Value, Max : Integer ) : Integer;
8810:   Function MirrorPow21( Value, Min, Max : Integer ) : Integer;
8811:   Function GetOptimalWrap( Max : Integer ) : TWrapProc;
8812:   Function GetOptimalWrap1( Min, Max : Integer ) : TWrapProcEx;
8813:   Function GetOptimalMirror( Max : Integer ) : TWrapProc;
8814:   Function GetOptimalMirror1( Min, Max : Integer ) : TWrapProcEx;
8815:   Function GetWrapProc( WrapMode : TWrapMode ) : TWrapProc;
8816:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer ) : TWrapProc;
8817:   Function GetWrapProcEx( WrapMode : TWrapMode ) : TWrapProcEx;
8818:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer ) : TWrapProcEx;
8819:   Function Div255( Value : Cardinal ) : Cardinal
8820:   Function SAR_4( Value : Integer ) : Integer
8821:   Function SAR_8( Value : Integer ) : Integer
8822:   Function SAR_9( Value : Integer ) : Integer
8823:   Function SAR_11( Value : Integer ) : Integer

```

```

8824: Function SAR_12( Value : Integer ) : Integer
8825: Function SAR_13( Value : Integer ) : Integer
8826: Function SAR_14( Value : Integer ) : Integer
8827: Function SAR_15( Value : Integer ) : Integer
8828: Function SAR_16( Value : Integer ) : Integer
8829: Function ColorSwap( WinColor : TColor ) : TColor32
8830: end;
8831:
8832: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8833: begin
8834: AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8835: Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8836: Procedure CopyComponents1(Dst:TCustomBitmap32;DstX,
DstY:Int32;Src:TCustomBitmap32;SrcRect:TRect;Components:TColor32Comp;
8837: Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32 )
8838: Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean )
8839: Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32 )
8840: Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components )
8841: Procedure InvertRGB( Dst, Src : TCustomBitmap32 )
8842: Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean )
8843: Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32 )
8844: Function CreateBitmask( Components : TColor32Components ) : TColor32
8845: Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
Bitmask : TColor32; LogicalOperator : TLogicalOperator );
8846: Procedure
ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8847: Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean )
8848: end;
8849:
8850:
8851: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8852: begin
8853: AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8854: AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8855: Function NtfsGetCompression( const FileName : string; var State : Short ) : Boolean;
8856: Function NtfsGetCompression1( const FileName : string ) : TFileCompressionState;
8857: Function NtfsSetCompression( const FileName : string; const State : Short ) : Boolean
8858: Procedure NtfsSetFileCompression(const FileName: string; const State: TFileCompressionState)
8859: Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8860: Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8861: Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8862: //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloc'
8863: //+'+tedRangeBuffer; MoreData : Boolean; end
8864: Function NtfsSetSparse( const FileName : string ) : Boolean
8865: Function NtfsZeroDataByHandle( const Handle: THandle; const First, Last : Int64 ) : Boolean
8866: Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64 ) : Boolean
8867: //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
Ranges:TNtfsAllocRanges):Boolean;
8868: //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
Index:Integer):TFileAllocatedRangeBuffer
8869: Function NtfsParseStreamsSupported( const Volume : string ) : Boolean
8870: Function NtfsGetSparse( const FileName : string ) : Boolean
8871: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD ) : Boolean
8872: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword ) : Boolean
8873: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8874: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD ) : Boolean
8875: Function NtfsReparsePointsSupported( const Volume : string ) : Boolean
8876: Function NtfsFileHasReparsePoint( const Path : string ) : Boolean
8877: Function NtfsIsFolderMountPoint( const Path : string ) : Boolean
8878: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char ) : Boolean
8879: Function NtfsMountVolume( const Volume : Char; const MountPoint : string ) : Boolean
8880: AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8881: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8882: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8883: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8884: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8885: Function NtfsRequestOpLock(Handle:THandle, Kind : TOpLock; Overlapped : TOverlapped) : Bool
8886: Function NtfsCreateJunctionPoint( const Source, Destination : string ) : Boolean
8887: Function NtfsDeleteJunctionPoint( const Source : string ) : Boolean
8888: Function NtfsGetJunctionPointDestination(const Source : string;var Destination:string):Bool
8889: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8890: +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8891: AddTypeS('TStreamIds', 'set of TStreamId
8892: AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8893: +': __Pointer; StreamIds : TStreamIds; end
8894: AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8895: +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8896: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8897: Function NtfsFindNextStream( var Data : TFindStreamData ) : Boolean
8898: Function NtfsFindStreamClose( var Data : TFindStreamData ) : Boolean
8899: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string ) : Boolean
8900: AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8901: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean
8902: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
List:TStrings):Bool;
8903: Function NtfsDeleteHardLinks( const FileName : string ) : Boolean
8904: Function JclAppInstances : TJclAppInstances;
8905: Function JclAppInstances1( const UniqueAppIdGuidStr : string ) : TJclAppInstances;
8906: Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND ) : TJclAppInstDataKind

```

```

8907: Procedure ReadMessageData( const Message: TMessage; var Data : __Pointer; var Size : Integer)
8908: Procedure ReadMessageString( const Message : TMessage; var S : string)
8909: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8910:
8911:
8912: (*-----*)
8913: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8914: begin
8915:   FindClass ('TOBJECT'), 'EJclGraphicsError
8916:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8917:   TDynPointArray', 'array of TPoint
8918:   TDynDynPointArrayArray', 'array of TDynPointArray
8919:   TPointF', 'record X : Single; Y : Single; end
8920:   TDynPointArrayF', 'array of TPointF
8921:   TDrawMode2', '( dmOpaque, dmBlend )
8922:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8923:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8924:   TResamplingFilter', '(rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8925:   TMatrix3d', 'record array[0..2,0..2] of extended end
8926:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8927:   TScanLine', 'array of Integer
8928:   TScanLines', 'array of TScanLine
8929:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8930:   TGradientDirection', '( gdVertical, gdHorizontal )
8931:   TPolyFillMode', '( fmAlternate, fmWinding )
8932:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8933:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8934:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8935:   SIRegister_TJclDesktopCanvas(CL);
8936:   FindClass ('TOBJECT'), 'TJclRegion
8937:   SIRegister_TJclRegionInfo(CL);
8938:   SIRegister_TJclRegion(CL);
8939:   SIRegister_TJclThreadPersistent(CL);
8940:   SIRegister_TJclCustomMap(CL);
8941:   SIRegister_TJclBitmap32(CL);
8942:   SIRegister_TJclByteMap(CL);
8943:   SIRegister_TJclTransformation(CL);
8944:   SIRegister_TJclLinearTransformation(CL);
8945:   Procedure Stretch(NewWidth,
NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8946:   Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8947:   Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer)
8948:   Function GetAntialiasedBitmap( const Bitmap : TBitmap ) : TBitmap
8949:   Procedure BitmapToJpeg( const FileName : string)
8950:   Procedure JpegToBitmap( const FileName : string)
8951:   Function ExtractIconCount( const FileName : string ) : Integer
8952:   Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer ) : HICON
8953:   Function IconToBitmapJ( Icon : HICON ) : HBITMAP
8954:   Procedure
BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8955:   Procedure StretchTransfer(Dst:TJclBitmap32;
DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8956:   Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8957:   Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8958:   Function FillGradient( DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection : TGradientDirection ) : Boolean;
8959:   Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
RegionBitmapMode:TJclRegionBitmapMode) : HRGN
8960:   Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND );
8961:   Procedure ScreenShot1( bm : TBitmap );
8962:   Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8963:   Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8964:   Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8965:   Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8966:   Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8967:   Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8968:   Procedure PolyPolygonTS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8969:   Procedure PolyPolygonAS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8970:   Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8971:   Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8972:   Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8973:   Procedure Invert( Dst, Src : TJclBitmap32)
8974:   Procedure InvertRGB( Dst, Src : TJclBitmap32)
8975:   Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8976:   Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8977:   Procedure SetGamma( Gamma : Single)
8978: end;
8979:
8980: (*-----*)
8981: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8982: begin
8983:   Function LockedAdd( var Target : Integer; Value : Integer ) : Integer
8984:   Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer ) : Integer
8985:   Function LockedCompareExchange1( var Target : __Pointer; Exch, Comp : __Pointer ) : Pointer;
8986:   Function LockedDec( var Target : Integer ) : Integer
8987:   Function LockedExchange( var Target : Integer; Value : Integer ) : Integer
8988:   Function LockedExchangeAdd( var Target : Integer; Value : Integer ) : Integer
8989:   Function LockedExchangeDec( var Target : Integer ) : Integer
8990:   Function LockedExchangeInc( var Target : Integer ) : Integer

```

```

8991: Function LockedExchangeSub( var Target : Integer; Value : Integer ) : Integer
8992: Function LockedInc( var Target : Integer ) : Integer
8993: Function LockedSub( var Target : Integer; Value : Integer ) : Integer
8994:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8995:   SIRegister_TJclDispatcherObject(CL);
8996: Function WaitForMultipleObjects(const Objects:array of
TJclDispatcherObject;WaitAll:Boolean;TimeOut:Cardinal):Cardinal;
8997: Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Boolean;
TimeOut : Cardinal):Cardinal
8998:   SIRegister_TJclCriticalSection(CL);
8999:   SIRegister_TJclCriticalSectionEx(CL);
9000:   SIRegister_TJclEvent(CL);
9001:   SIRegister_TJclWaitableTimer(CL);
9002:   SIRegister_TJclSemaphore(CL);
9003:   SIRegister_TJclMutex(CL);
9004:   POptexSharedInfo', '^POptexSharedInfo // will not work
9005:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
9006:   SIRegister_TJclOptex(CL);
9007:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
9008:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
9009:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
9010:   SIRegister_TJclMultiReadExclusiveWrite(CL);
9011:   PMetSectSharedInfo', '^PMetSectSharedInfo // will not work
9012:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
9013:     +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
9014:   PMeteredSection', '^TMeteredSection // will not work
9015:   TMeteredSection', 'record Event:THandle; FileMap:THandle; SharedInfo:PMetSectSharedInfo; end
9016:   SIRegister_TJclMeteredSection(CL);
9017:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
9018:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
9019:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
9020:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
9021: Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
9022: Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
9023: Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
9024: Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
9025: Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
9026:   FindClass('TOBJECT'), 'EJclWin32HandleObjectError
9027:   FindClass('TOBJECT'), 'EJclDispatcherObjectError
9028:   FindClass('TOBJECT'), 'EJclCriticalSectionError
9029:   FindClass('TOBJECT'), 'EJclEventError
9030:   FindClass('TOBJECT'), 'EJclWaitableTimerError
9031:   FindClass('TOBJECT'), 'EJclSemaphoreError
9032:   FindClass('TOBJECT'), 'EJclMutexError
9033:   FindClass('TOBJECT'), 'EJclMeteredSectionError
9034: end;
9035:
9036:
9037: //*****unit uPSI_mORMotReport;
9038: Procedure SetCurrentPrinterAsDefault
9039: Function CurrentPrinterName : string
9040: Function mCurrentPrinterPaperSize : string
9041: Procedure UseDefaultPrinter
9042:
9043: procedure SIRegisterTSTREAM(C1: TPSFascalCompiler);
9044: begin
9045:   with FindClass('TOBJECT'), 'TStream') do begin
9046:     IsAbstract := True;
9047:     //RegisterMethod('Function Read( var Buffer, Count : Longint ) : Longint
9048:     // RegisterMethod('Function Write( const Buffer, Count : Longint ) : Longint
9049:       function Read(Buffer:String;Count:LongInt):LongInt
9050:       function Write(Buffer:String;Count:LongInt):LongInt
9051:       function ReadString(Buffer:String;Count:LongInt):LongInt //FileStream
9052:       function WriteString(Buffer:String;Count:LongInt):LongInt
9053:       function ReadInt(Buffer:integer;Count:LongInt):LongInt
9054:       function WriteInt(Buffer:integer;Count:LongInt):LongInt
9055:       function ReadByteArray(Buffer:TByteToArray;Count:LongInt):LongInt';
9056:       function WriteByteArray(Buffer:TByteToArray;Count:LongInt):LongInt';
9057:
9058:       procedure ReadAB(Buffer: TByteToArray;Count:LongInt)
9059:       procedure WriteAB(Buffer: TByteToArray;Count:LongInt)
9060:       procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
9061:       procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
9062:       procedure ReadAC(Buffer: TCharArray;Count:LongInt)
9063:       procedure WriteAC(Buffer: TCharArray;Count:LongInt)
9064:       procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
9065:       procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
9066:
9067:       function Seek(Offset:LongInt;Origin:Word):LongInt
9068:       procedure ReadBuffer(Buffer:String;Count:LongInt)
9069:       procedure WriteBuffer(Buffer:String;Count:LongInt)
9070:       procedure ReadBufferInt(Buffer:Integer;Count:LongInt');
9071:       procedure WriteBufferInt(Buffer:Integer;Count:LongInt');
9072:       procedure ReadBufferFloat(Buffer:Double;Count:LongInt');
9073:       Procedure WriteBufferFloat(Buffer:Double;Count:LongInt');
9074:
9075:       procedure ReadBufferAB(Buffer: TByteToArray;Count:LongInt)
9076:       procedure WriteBufferAB(Buffer: TByteToArray;Count:LongInt)
9077:       procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)

```

```

9078:   procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9079:   procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9080:   procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9081:   procedure ReadBufferAWD(Buffer: TTWordDynArray;Count:LongInt)
9082:   procedure WriteBufferAWD(Buffer: TTWordDynArray;Count:LongInt)
9083:   procedure ReadBufferAW(Buffer: TTWordArray;Count:LongInt)
9084:   procedure WriteBufferAW(Buffer: TTWordArray;Count:LongInt)
9085:   procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9086:   procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9087:   procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9088:   procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9089:
9090:   procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9091:   procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9092:   procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
9093:   procedure WriteBufferO(Buffer: TObject;Count:LongInt)';
9094: //READBUFFERAC
9095: function InstanceSize: Longint
9096: Procedure FixupResourceHeader( FixupInfo : Integer)
9097: Procedure ReadResHeader
9098:
9099: {$IFDEF DELPHI4UP}
9100: function CopyFrom(Source:TStream;Count:Int64):LongInt
9101: {$ELSE}
9102: function CopyFrom(Source:TStream;Count:Integer):LongInt
9103: {$ENDIF}
9104: RegisterProperty('Position', 'LongInt', iptrw);
9105: RegisterProperty('Size', 'LongInt', iptrw);
9106: end;
9107: end;
9108:
9109:
9110: { **** -----
9111:   Unit DMATH - Interface for DMATH.DLL
9112:   ----- }
9113: // see more docs/dmath_manual.pdf
9114:
9115: Function InitEval : Integer
9116: Procedure SetVariable( VarName : Char; Value : Float)
9117: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9118: Function Eval( ExpressionString : String) : Float
9119:
9120: unit dmath; //types are in built, others are external in DLL
9121: interface
9122: {$IFDEF DELPHI}
9123: uses
9124:   StdCtrls, Graphics;
9125: {$ENDIF}
9126: { -----
9127:   Types and constants
9128:   ----- }
9129: {$i types.inc}
9130: { -----
9131:   Error handling
9132:   ----- }
9133: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9134: { Sets the error code }
9135: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9136: { Sets error code and default function value }
9137: function MathErr : Integer; external 'dmath';
9138: { Returns the error code }
9139: { -----
9140:   Dynamic arrays
9141:   ----- }
9142: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9143: { Sets the auto-initialization of arrays }
9144: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9145: { Creates floating point vector V[0..Ub] }
9146: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9147: { Creates integer vector V[0..Ub] }
9148: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9149: { Creates complex vector V[0..Ub] }
9150: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9151: { Creates boolean vector V[0..Ub] }
9152: procedure DimStrVector(var V : TStringVector; Ub : Integer); external 'dmath';
9153: { Creates string vector V[0..Ub] }
9154: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9155: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9156: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9157: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9158: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9159: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9160: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9161: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9162: procedure DimStrMatrix(var A : TStringMatrix; Ub1, Ub2 : Integer); external 'dmath';
9163: { Creates string matrix A[0..Ub1, 0..Ub2] }
9164: { -----
9165:   Minimum, maximum, sign and exchange
9166:   ----- }

```

```

9167: function FMin(X, Y : Float) : Float; external 'dmath';
9168: { Minimum of 2 reals }
9169: function FMax(X, Y : Float) : Float; external 'dmath';
9170: { Maximum of 2 reals }
9171: function IMin(X, Y : Integer) : Integer; external 'dmath';
9172: { Minimum of 2 integers }
9173: function IMax(X, Y : Integer) : Integer; external 'dmath';
9174: { Maximum of 2 integers }
9175: function Sgn(X : Float) : Integer; external 'dmath';
9176: { Sign (returns 1 if X = 0) }
9177: function Sgn0(X : Float) : Integer; external 'dmath';
9178: { Sign (returns 0 if X = 0) }
9179: function DSgn(A, B : Float) : Float; external 'dmath';
9180: { Sgn(B) * |A| }
9181: procedure FSwap(var X, Y : Float); external 'dmath';
9182: { Exchange 2 reals }
9183: procedure ISwap(var X, Y : Integer); external 'dmath';
9184: { Exchange 2 integers }
9185: { -----
9186:   Rounding functions
9187: ----- }
9188: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9189: { Rounds X to N decimal places }
9190: function Ceil(X : Float) : Integer; external 'dmath';
9191: { Ceiling function }
9192: function Floor(X : Float) : Integer; external 'dmath';
9193: { Floor function }
9194: { -----
9195:   Logarithms, exponentials and power
9196: ----- }
9197: function Expo(X : Float) : Float; external 'dmath';
9198: { Exponential }
9199: function Exp2(X : Float) : Float; external 'dmath';
9200: { 2^X }
9201: function Exp10(X : Float) : Float; external 'dmath';
9202: { 10^X }
9203: function Log(X : Float) : Float; external 'dmath';
9204: { Natural log }
9205: function Log2(X : Float) : Float; external 'dmath';
9206: { Log, base 2 }
9207: function Log10(X : Float) : Float; external 'dmath';
9208: { Decimal log }
9209: function LogA(X, A : Float) : Float; external 'dmath';
9210: { Log, base A }
9211: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9212: { X^N }
9213: function Power(X, Y : Float) : Float; external 'dmath';
9214: { X^Y, X >= 0 }
9215: { -----
9216:   Trigonometric functions
9217: ----- }
9218: function Pythag(X, Y : Float) : Float; external 'dmath';
9219: { Sqrt(X^2 + Y^2) }
9220: function FixAngle(Theta : Float) : Float; external 'dmath';
9221: { Set Theta in -Pi..Pi }
9222: function Tan(X : Float) : Float; external 'dmath';
9223: { Tangent }
9224: function ArcSin(X : Float) : Float; external 'dmath';
9225: { Arc sinus }
9226: function ArcCos(X : Float) : Float; external 'dmath';
9227: { Arc cosinus }
9228: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9229: { Angle (Ox, OM) with M(X,Y) }
9230: { -----
9231:   Hyperbolic functions
9232: ----- }
9233: function Sinh(X : Float) : Float; external 'dmath';
9234: { Hyperbolic sine }
9235: function Cosh(X : Float) : Float; external 'dmath';
9236: { Hyperbolic cosine }
9237: function Tanh(X : Float) : Float; external 'dmath';
9238: { Hyperbolic tangent }
9239: function ArcSinh(X : Float) : Float; external 'dmath';
9240: { Inverse hyperbolic sine }
9241: function ArcCosh(X : Float) : Float; external 'dmath';
9242: { Inverse hyperbolic cosine }
9243: function ArcTanh(X : Float) : Float; external 'dmath';
9244: { Inverse hyperbolic tangent }
9245: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9246: { Sinh & Cosh }
9247: { -----
9248:   Gamma function and related functions
9249: ----- }
9250: function Fact(N : Integer) : Float; external 'dmath';
9251: { Factorial }
9252: function SgnGamma(X : Float) : Integer; external 'dmath';
9253: { Sign of Gamma function }
9254: function Gamma(X : Float) : Float; external 'dmath';
9255: { Gamma function }

```

```

9256: function LnGamma(X : Float) : Float; external 'dmath';
9257: { Logarithm of Gamma function }
9258: function Stirling(X : Float) : Float; external 'dmath';
9259: { Stirling's formula for the Gamma function }
9260: function StirLog(X : Float) : Float; external 'dmath';
9261: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9262: function DiGamma(X : Float) : Float; external 'dmath';
9263: { Digamma function }
9264: function TriGamma(X : Float) : Float; external 'dmath';
9265: { Trigamma function }
9266: function IGamma(A, X : Float) : Float; external 'dmath';
9267: { Incomplete Gamma function }
9268: function JGamma(A, X : Float) : Float; external 'dmath';
9269: { Complement of incomplete Gamma function }
9270: function InvGamma(A, P : Float) : Float; external 'dmath';
9271: { Inverse of incomplete Gamma function }
9272: function Erf(X : Float) : Float; external 'dmath';
9273: { Error function }
9274: function Erfc(X : Float) : Float; external 'dmath';
9275: { Complement of error function }
9276: { -----
9277: Beta function and related functions
9278: ----- }
9279: function Beta(X, Y : Float) : Float; external 'dmath';
9280: { Beta function }
9281: function IBeta(A, B, X : Float) : Float; external 'dmath';
9282: { Incomplete Beta function }
9283: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9284: { Inverse of incomplete Beta function }
9285: { -----
9286: Lambert's function
9287: ----- }
9288: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9289: { -----
9290: Binomial distribution
9291: ----- }
9292: function Binomial(N, K : Integer) : Float; external 'dmath';
9293: { Binomial coefficient C(N,K) }
9294: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9295: { Probability of binomial distribution }
9296: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9297: { Cumulative probability for binomial distrib. }
9298: { -----
9299: Poisson distribution
9300: ----- }
9301: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9302: { Probability of Poisson distribution }
9303: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9304: { Cumulative probability for Poisson distrib. }
9305: { -----
9306: Exponential distribution
9307: ----- }
9308: function DExpo(A, X : Float) : Float; external 'dmath';
9309: { Density of exponential distribution with parameter A }
9310: function FExpo(A, X : Float) : Float; external 'dmath';
9311: { Cumulative probability function for exponential dist. with parameter A }
9312: { -----
9313: Standard normal distribution
9314: ----- }
9315: function DNorm(X : Float) : Float; external 'dmath';
9316: { Density of standard normal distribution }
9317: function FNorm(X : Float) : Float; external 'dmath';
9318: { Cumulative probability for standard normal distrib. }
9319: function PNorm(X : Float) : Float; external 'dmath';
9320: { Prob(|U| > X) for standard normal distrib. }
9321: function InvNorm(P : Float) : Float; external 'dmath';
9322: { Inverse of standard normal distribution }
9323: { -----
9324: Student's distribution
9325: ----- }
9326: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9327: { Density of Student distribution with Nu d.o.f. }
9328: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9329: { Cumulative probability for Student distrib. with Nu d.o.f. }
9330: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9331: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9332: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9333: { Inverse of Student's t-distribution function }
9334: { -----
9335: Khi-2 distribution
9336: ----- }
9337: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9338: { Density of Khi-2 distribution with Nu d.o.f. }
9339: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9340: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9341: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9342: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9343: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9344: { Inverse of Khi-2 distribution function }

```

```

9345: { -----
9346:   Fisher-Snedecor distribution
9347:   -----
9348:   function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9349:   { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9350:   function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9351:   { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9352:   function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9353:   { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9354:   function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9355:   { Inverse of Snedecor's F-distribution function }
9356:   -----
9357:   Beta distribution
9358:   -----
9359:   function DBeta(A, B, X : Float) : Float; external 'dmath';
9360:   { Density of Beta distribution with parameters A and B }
9361:   function FBeta(A, B, X : Float) : Float; external 'dmath';
9362:   { Cumulative probability for Beta distrib. with param. A and B }
9363:   -----
9364:   Gamma distribution
9365:   -----
9366:   function DGamma(A, B, X : Float) : Float; external 'dmath';
9367:   { Density of Gamma distribution with parameters A and B }
9368:   function FGamma(A, B, X : Float) : Float; external 'dmath';
9369:   { Cumulative probability for Gamma distrib. with param. A and B }
9370:   -----
9371:   Expression evaluation
9372:   -----
9373:   function InitEval : Integer; external 'dmath';
9374:   { Initializes built-in functions and returns their number }
9375:   function Eval(ExpressionString : String) : Float; external 'dmath';
9376:   { Evaluates an expression at run-time }
9377:   procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9378:   { Assigns a value to a variable }
9379:   procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9380:   { Adds a function to the parser }
9381:   -----
9382:   Matrices and linear equations
9383:   -----
9384:   procedure GaussJordan(A          : TMatrix;
9385:                         Lb, Ub1, Ub2 : Integer;
9386:                         var Det      : Float); external 'dmath';
9387:   { Transforms a matrix according to the Gauss-Jordan method }
9388:   procedure LinEq(A          : TMatrix;
9389:                     B          : TVector;
9390:                     Lb, Ub : Integer;
9391:                     var Det : Float); external 'dmath';
9392:   { Solves a linear system according to the Gauss-Jordan method }
9393:   procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9394:   { Cholesky factorization of a positive definite symmetric matrix }
9395:   procedure LU_Decompo(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9396:   { LU decomposition }
9397:   procedure LU_Solve(A       : TMatrix;
9398:                      B       : TVector;
9399:                      Lb, Ub : Integer;
9400:                      X       : TVector); external 'dmath';
9401:   { Solution of linear system from LU decomposition }
9402:   procedure QR_Decompo(A       : TMatrix;
9403:                         Lb, Ub1, Ub2 : Integer;
9404:                         R       : TMatrix); external 'dmath';
9405:   { QR decomposition }
9406:   procedure QR_Solve(Q, R       : TMatrix;
9407:                      B       : TVector;
9408:                      Lb, Ub1, Ub2 : Integer;
9409:                      X       : TVector); external 'dmath';
9410:   { Solution of linear system from QR decomposition }
9411:   procedure SV_Decompo(A       : TMatrix;
9412:                         Lb, Ub1, Ub2 : Integer;
9413:                         S       : TVector;
9414:                         V       : TMatrix); external 'dmath';
9415:   { Singular value decomposition }
9416:   procedure SV_SetZero(S       : TVector;
9417:                         Lb, Ub : Integer;
9418:                         Tol    : Float); external 'dmath';
9419:   { Set lowest singular values to zero }
9420:   procedure SV_Solve(U       : TMatrix;
9421:                         S       : TVector;
9422:                         V       : TMatrix;
9423:                         B       : TVector;
9424:                         Lb, Ub1, Ub2 : Integer;
9425:                         X       : TVector); external 'dmath';
9426:   { Solution of linear system from SVD }
9427:   procedure SV_Approx(U       : TMatrix;
9428:                         S       : TVector;
9429:                         V       : TMatrix;
9430:                         Lb, Ub1, Ub2 : Integer;
9431:                         A       : TMatrix); external 'dmath';
9432:   { Matrix approximation from SVD }
9433:   procedure EigenVals(A       : TMatrix;

```

```

9434:           Lb, Ub : Integer;
9435:           Lambda : TCompVector); external 'dmath';
9436: { Eigenvalues of a general square matrix }
9437: procedure EigenVect(A : TMatrix;
9438:           Lb, Ub : Integer;
9439:           Lambda : TCompVector;
9440:           V : TMatrix); external 'dmath';
9441: { Eigenvalues and eigenvectors of a general square matrix }
9442: procedure EigenSym(A : TMatrix;
9443:           Lb, Ub : Integer;
9444:           Lambda : TVector;
9445:           V : TMatrix); external 'dmath';
9446: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9447: procedure Jacobi(A : TMatrix;
9448:           Lb, Ub, MaxIter : Integer;
9449:           Tol : Float;
9450:           Lambda : TVector;
9451:           V : TMatrix); external 'dmath';
9452: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9453: { -----
9454: Optimization
9455: ----- }
9456: procedure MinBrack(Func : TFunc;
9457:           var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9458: { Brackets a minimum of a function }
9459: procedure GoldSearch(Func : TFunc;
9460:           A, B : Float;
9461:           MaxIter : Integer;
9462:           Tol : Float;
9463:           var Xmin, Ymin : Float); external 'dmath';
9464: { Minimization of a function of one variable (golden search) }
9465: procedure LinMin(Func : TFuncNVar;
9466:           X, DeltaX : TVector;
9467:           Lb, Ub : Integer;
9468:           var R : Float;
9469:           MaxIter : Integer;
9470:           Tol : Float;
9471:           var F_min : Float); external 'dmath';
9472: { Minimization of a function of several variables along a line }
9473: procedure Newton(Func : TFuncNVar;
9474:           HessGrad : THessGrad;
9475:           X : TVector;
9476:           Lb, Ub : Integer;
9477:           MaxIter : Integer;
9478:           Tol : Float;
9479:           var F_min : Float;
9480:           G : TVector;
9481:           H_inv : TMatrix;
9482:           var Det : Float); external 'dmath';
9483: { Minimization of a function of several variables (Newton's method) }
9484: procedure SaveNewton(FileName : string); external 'dmath';
9485: { Save Newton iterations in a file }
9486: procedure Marquardt(Func : TFuncNVar;
9487:           HessGrad : THessGrad;
9488:           X : TVector;
9489:           Lb, Ub : Integer;
9490:           MaxIter : Integer;
9491:           Tol : Float;
9492:           var F_min : Float;
9493:           G : TVector;
9494:           H_inv : TMatrix;
9495:           var Det : Float); external 'dmath';
9496: { Minimization of a function of several variables (Marquardt's method) }
9497: procedure SaveMarquardt(FileName : string); external 'dmath';
9498: { Save Marquardt iterations in a file }
9499: procedure BFGS(Func : TFuncNVar;
9500:           Gradient : TGradient;
9501:           X : TVector;
9502:           Lb, Ub : Integer;
9503:           MaxIter : Integer;
9504:           Tol : Float;
9505:           var F_min : Float;
9506:           G : TVector;
9507:           H_inv : TMatrix); external 'dmath';
9508: { Minimization of a function of several variables (BFGS method) }
9509: procedure SaveBFGS(FileName : string); external 'dmath';
9510: { Save BFGS iterations in a file }
9511: procedure Simplex(Func : TFuncNVar;
9512:           X : TVector;
9513:           Lb, Ub : Integer;
9514:           MaxIter : Integer;
9515:           Tol : Float;
9516:           var F_min : Float); external 'dmath';
9517: { Minimization of a function of several variables (Simplex) }
9518: procedure SaveSimplex(FileName : string); external 'dmath';
9519: { Save Simplex iterations in a file }
9520: { -----
9521: Nonlinear equations
9522: ----- }
```

```

9523: procedure RootBrack(Func : TFunc;
9524:           var X, Y, FX, FY : Float); external 'dmath';
9525: { Brackets a root of function Func between X and Y }
9526: procedure Bisect(Func : TFunc;
9527:           var X, Y : Float;
9528:           MaxIter : Integer;
9529:           Tol : Float;
9530:           var F : Float); external 'dmath';
9531: { Bisection method }
9532: procedure Secant(Func : TFunc;
9533:           var X, Y : Float;
9534:           MaxIter : Integer;
9535:           Tol : Float;
9536:           var F : Float); external 'dmath';
9537: { Secant method }
9538: procedure NewtEq(Func, Deriv : TFunc;
9539:           var X : Float;
9540:           MaxIter : Integer;
9541:           Tol : Float;
9542:           var F : Float); external 'dmath';
9543: { Newton-Raphson method for a single nonlinear equation }
9544: procedure NewtEqs(Equations : TEquations;
9545:           Jacobian : TJacobian;
9546:           X, F : TVector;
9547:           Lb, Ub : Integer;
9548:           MaxIter : Integer;
9549:           Tol : Float); external 'dmath';
9550: { Newton-Raphson method for a system of nonlinear equations }
9551: procedure Broyden(Equations : TEquations;
9552:           X, F : TVector;
9553:           Lb, Ub : Integer;
9554:           MaxIter : Integer;
9555:           Tol : Float); external 'dmath';
9556: { Broyden's method for a system of nonlinear equations }
9557: { -----
9558:   Polynomials and rational fractions
9559: ----- }
9560: function Poly(X : Float;
9561:           Coef : TVector;
9562:           Deg : Integer) : Float; external 'dmath';
9563: { Evaluates a polynomial }
9564: function RFrac(X : Float;
9565:           Coef : TVector;
9566:           Deg1, Deg2 : Integer) : Float; external 'dmath';
9567: { Evaluates a rational fraction }
9568: function RootPol1(A, B : Float;
9569:           var X : Float) : Integer; external 'dmath';
9570: { Solves the linear equation A + B * X = 0 }
9571: function RootPol2(Coef : TVector;
9572:           Z : TCompVector) : Integer; external 'dmath';
9573: { Solves a quadratic equation }
9574: function RootPol3(Coef : TVector;
9575:           Z : TCompVector) : Integer; external 'dmath';
9576: { Solves a cubic equation }
9577: function RootPol4(Coef : TVector;
9578:           Z : TCompVector) : Integer; external 'dmath';
9579: { Solves a quartic equation }
9580: function RootPol(Coef : TVector;
9581:           Deg : Integer;
9582:           Z : TCompVector) : Integer; external 'dmath';
9583: { Solves a polynomial equation }
9584: function SetRealRoots(Deg : Integer;
9585:           Z : TCompVector;
9586:           Tol : Float) : Integer; external 'dmath';
9587: { Set the imaginary part of a root to zero }
9588: procedure SortRoots(Deg : Integer;
9589:           Z : TCompVector); external 'dmath';
9590: { Sorts the roots of a polynomial }
9591: { -----
9592:   Numerical integration and differential equations
9593: ----- }
9594: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9595: { Integration by trapezoidal rule }
9596: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9597: { Integral from A to B }
9598: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9599: { Integral from 0 to B }
9600: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9601: { Convolution product at time T }
9602: procedure ConvTrap(Func1, Func2 : TFunc; T, Y : TVector; N : Integer); external 'dmath';
9603: { Convolution by trapezoidal rule }
9604: procedure RKF45(F : TDiffEqs;
9605:           Neqn : Integer;
9606:           Y, Yp : TVector;
9607:           var T : Float;
9608:           Tout, RelErr, AbsErr : Float;
9609:           var Flag : Integer); external 'dmath';
9610: { Integration of a system of differential equations }
9611: { ----- }

```

```

9612: Fast Fourier Transform
9613: -----
9614: procedure FFT(NumSamples : Integer;
9615:           InArray, OutArray : TCompVector); external 'dmath';
9616: { Fast Fourier Transform }
9617: procedure IFFT(NumSamples : Integer;
9618:           InArray, OutArray : TCompVector); external 'dmath';
9619: { Inverse Fast Fourier Transform }
9620: procedure FFT_Integer(NumSamples : Integer;
9621:           RealIn, ImagIn : TIntVector;
9622:           OutArray : TCompVector); external 'dmath';
9623: { Fast Fourier Transform for integer data }
9624: procedure FFT_Integer_Cleanup; external 'dmath';
9625: { Clear memory after a call to FFT_Integer }
9626: procedure CalcFrequency(NumSamples,
9627:           FrequencyIndex : Integer;
9628:           InArray : TCompVector;
9629:           var FFT : Complex); external 'dmath';
9630: { Direct computation of Fourier transform }
9631: { -----
9632:   Random numbers
9633: -----
9634: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9635: { Select generator }
9636: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9637: { Initialize generator }
9638: function IRanGen : RNG_IntType; external 'dmath';
9639: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9640: function IRanGen31 : RNG_IntType; external 'dmath';
9641: { 31-bit random integer in [0 .. 2^31 - 1] }
9642: function RanGen1 : Float; external 'dmath';
9643: { 32-bit random real in {0,1} }
9644: function RanGen2 : Float; external 'dmath';
9645: { 32-bit random real in {0,1} }
9646: function RanGen3 : Float; external 'dmath';
9647: { 32-bit random real in (0,1) }
9648: function RanGen53 : Float; external 'dmath';
9649: { 53-bit random real in {0,1} }
9650: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9651: { Initializes the 'Multiply with carry' random number generator }
9652: function IRanMWC : RNG_IntType; external 'dmath';
9653: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9654: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9655: { Initializes Mersenne Twister generator with a seed }
9656: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9657:           KeyLength : Word); external 'dmath';
9658: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9659: function IRanMT : RNG_IntType; external 'dmath';
9660: { Random integer from MT generator }
9661: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9662: { Initializes the UVAG generator with a string }
9663: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9664: { Initializes the UVAG generator with an integer }
9665: function IRanUVAG : RNG_IntType; external 'dmath';
9666: { Random integer from UVAG generator }
9667: function RanGaussStd : Float; external 'dmath';
9668: { Random number from standard normal distribution }
9669: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9670: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9671: procedure RanMult(M : TVector; L : TMatrix;
9672:           Lb, Ub : Integer;
9673:           X : TVector); external 'dmath';
9674: { Random vector from multinormal distribution (correlated) }
9675: procedure RanMultIndep(M, S : TVector;
9676:           Lb, Ub : Integer;
9677:           X : TVector); external 'dmath';
9678: { Random vector from multinormal distribution (uncorrelated) }
9679: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9680: { Initializes Metropolis-Hastings parameters }
9681: procedure GetMHParams(var NCycles, MaxSim, SavedSim:Integer); external 'dmath';
9682: { Returns Metropolis-Hastings parameters }
9683: procedure Hastings(Func : TFuncNVar;
9684:           T : Float;
9685:           X : TVector;
9686:           V : TMatrix;
9687:           Lb, Ub : Integer;
9688:           Xmat : TMatrix;
9689:           X_min : TVector;
9690:           var F_min : Float); external 'dmath';
9691: { Simulation of a probability density function by Metropolis-Hastings }
9692: procedure InitsSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9693: { Initializes Simulated Annealing parameters }
9694: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9695: { Initializes log file }
9696: procedure SimAnn(Func : TFuncNVar;
9697:           X, Xmin, Xmax : TVector;
9698:           Lb, Ub : Integer;
9699:           var F_min : Float); external 'dmath';
9700: { Minimization of a function of several var. by simulated annealing }

```

```

9701: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9702: { Initializes Genetic Algorithm parameters }
9703: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9704: { Initializes log file }
9705: procedure GenAlg(Func : TFuncNVar;
9706:   X, Xmin, Xmax : TVector;
9707:   Lb, Ub : Integer;
9708:   var F_min : Float); external 'dmath';
9709: { Minimization of a function of several var. by genetic algorithm }
9710: { -----
9711:   Statistics
9712:   ----- }
9713: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9714: { Mean of sample X }
9715: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9716: { Minimum of sample X }
9717: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9718: { Maximum of sample X }
9719: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9720: { Median of sample X }
9721: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9722: { Standard deviation estimated from sample X }
9723: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9724: { Standard deviation of population }
9725: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9726: { Correlation coefficient }
9727: function Skewness(X : TVector; Lb, Ub : Integer; M, Sigma: Float): Float; external 'dmath';
9728: { Skewness of sample X }
9729: function Kurtosis(X : TVector; Lb, Ub : Integer; M, Sigma: Float): Float; external 'dmath';
9730: { Kurtosis of sample X }
9731: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9732: { Quick sort (ascending order) }
9733: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9734: { Quick sort (descending order) }
9735: procedure Interval(X1, X2 : Float;
9736:   MinDiv, MaxDiv : Integer;
9737:   var Min, Max, Step : Float); external 'dmath';
9738: { Determines an interval for a set of values }
9739: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9740:   var XMin, XMax, XStep : Float); external 'dmath';
9741: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9742: procedure StudIndep(N1, N2 : Integer;
9743:   M1, M2, S1, S2 : Float;
9744:   var T : Float;
9745:   var DoF : Integer); external 'dmath';
9746: { Student t-test for independent samples }
9747: procedure StudPaired(X, Y : TVector;
9748:   Lb, Ub : Integer;
9749:   var T : Float;
9750:   var DoF : Integer); external 'dmath';
9751: { Student t-test for paired samples }
9752: procedure AnOVal(Ns : Integer;
9753:   N : TIntVector;
9754:   M, S : TVector;
9755:   var V_f, V_r, F : Float;
9756:   var DoF_f, DoF_r : Integer); external 'dmath';
9757: { One-way analysis of variance }
9758: procedure AnOVA2(NA, NB, Nobs : Integer;
9759:   M, S : TMatrix;
9760:   V, F : TVector;
9761:   DoF : TIntVector); external 'dmath';
9762: { Two-way analysis of variance }
9763: procedure Snedecor(N1, N2 : Integer;
9764:   S1, S2 : Float;
9765:   var F : Float;
9766:   var DoF1, DoF2 : Integer); external 'dmath';
9767: { Snedecor's F-test (comparison of two variances) }
9768: procedure Bartlett(Ns : Integer;
9769:   N : TIntVector;
9770:   S : TVector;
9771:   var Khi2 : Float;
9772:   var DoF : Integer); external 'dmath';
9773: { Bartlett's test (comparison of several variances) }
9774: procedure Mann_Whitney(N1, N2 : Integer;
9775:   X1, X2 : TVector;
9776:   var U, Eps : Float); external 'dmath';
9777: { Mann-Whitney test }
9778: procedure Wilcoxon(X, Y : TVector;
9779:   Lb, Ub : Integer;
9780:   var Ndiff : Integer;
9781:   var T, Eps : Float); external 'dmath';
9782: { Wilcoxon test }
9783: procedure Kruskal_Wallis(Ns : Integer;
9784:   N : TIntVector;
9785:   X : TMatrix;
9786:   var H : Float;
9787:   var DoF : Integer); external 'dmath';
9788: { Kruskal-Wallis test }
9789: procedure Khi2_Conform(N_cls : Integer;

```

```

9790:           N_estim : Integer;
9791:           Obs    : TIntVector;
9792:           Calc   : TVector;
9793:           var Khi2 : Float;
9794:           var DoF  : Integer); external 'dmath';
9795: { Khi-2 test for conformity }
9796: procedure Khi2_Indep(N_lin   : Integer;
9797:                       N_col   : Integer;
9798:                       Obs     : TIntMatrix;
9799:                       var Khi2 : Float;
9800:                       var DoF  : Integer); external 'dmath';
9801: { Khi-2 test for independence }
9802: procedure Woolf_Conform(N_cls  : Integer;
9803:                           N_estim : Integer;
9804:                           Obs     : TIntVector;
9805:                           Calc   : TVector;
9806:                           var G   : Float;
9807:                           var DoF  : Integer); external 'dmath';
9808: { Woolf's test for conformity }
9809: procedure Woolf_Indep(N_lin   : Integer;
9810:                         N_col   : Integer;
9811:                         Obs     : TIntMatrix;
9812:                         var G   : Float;
9813:                         var DoF  : Integer); external 'dmath';
9814: { Woolf's test for independence }
9815: procedure DimStatClassVector(var C : TStatClassVector;
9816:                                 Ub   : Integer); external 'dmath';
9817: { Allocates an array of statistical classes: C[0..Ub] }
9818: procedure Distrib(X      : TVector;
9819:                      Lb, Ub : Integer;
9820:                      A, B, H : Float;
9821:                      C      : TStatClassVector); external 'dmath';
9822: { Distributes an array X[Lb..Ub] into statistical classes }
9823: { -----
9824:   Linear / polynomial regression
9825: ----- }
9826: procedure LinFit(X, Y   : TVector;
9827:                      Lb, Ub : Integer;
9828:                      B      : TVector;
9829:                      V      : TMATRIX); external 'dmath';
9830: { Linear regression : Y = B(0) + B(1) * X }
9831: procedure WLinFit(X, Y, S : TVector;
9832:                      Lb, Ub : Integer;
9833:                      B      : TVector;
9834:                      V      : TMATRIX); external 'dmath';
9835: { Weighted linear regression : Y = B(0) + B(1) * X }
9836: procedure SVDFlinFit(X, Y   : TVector;
9837:                         Lb, Ub : Integer;
9838:                         SVDTol : Float;
9839:                         B      : TVector;
9840:                         V      : TMATRIX); external 'dmath';
9841: { Unweighted linear regression by singular value decomposition }
9842: procedure WSVDFlinFit(X, Y, S : TVector;
9843:                          Lb, Ub : Integer;
9844:                          SVDTol : Float;
9845:                          B      : TVector;
9846:                          V      : TMATRIX); external 'dmath';
9847: { Weighted linear regression by singular value decomposition }
9848: procedure MulFit(X      : TMATRIX;
9849:                      Y      : TVector;
9850:                      Lb, Ub, Nvar : Integer;
9851:                      ConsTerm : Boolean;
9852:                      B      : TVector;
9853:                      V      : TMATRIX); external 'dmath';
9854: { Multiple linear regression by Gauss-Jordan method }
9855: procedure WMulFit(X      : TMATRIX;
9856:                      Y, S   : TVector;
9857:                      Lb, Ub, Nvar : Integer;
9858:                      ConsTerm : Boolean;
9859:                      B      : TVector;
9860:                      V      : TMATRIX); external 'dmath';
9861: { Weighted multiple linear regression by Gauss-Jordan method }
9862: procedure SVDFit(X      : TMATRIX;
9863:                      Y      : TVector;
9864:                      Lb, Ub, Nvar : Integer;
9865:                      ConsTerm : Boolean;
9866:                      SVDTol : Float;
9867:                      B      : TVector;
9868:                      V      : TMATRIX); external 'dmath';
9869: { Multiple linear regression by singular value decomposition }
9870: procedure WSVDFit(X      : TMATRIX;
9871:                      Y, S   : TVector;
9872:                      Lb, Ub, Nvar : Integer;
9873:                      ConsTerm : Boolean;
9874:                      SVDTol : Float;
9875:                      B      : TVector;
9876:                      V      : TMATRIX); external 'dmath';
9877: { Weighted multiple linear regression by singular value decomposition }
9878: procedure PolFit(X, Y      : TVector;

```

```

9879:           Lb, Ub, Deg : Integer;
9880:           B          : TVector;
9881:           V          : TMatrix); external 'dmath';
9882: { Polynomial regression by Gauss-Jordan method }
9883: procedure WPolFit(X, Y, S : TVector;
9884:                      Lb, Ub, Deg : Integer;
9885:                      B          : TVector;
9886:                      V          : TMatrix); external 'dmath';
9887: { Weighted polynomial regression by Gauss-Jordan method }
9888: procedure SVDPolFit(X, Y : TVector;
9889:                        Lb, Ub, Deg : Integer;
9890:                        SVDTol   : Float;
9891:                        B          : TVector;
9892:                        V          : TMatrix); external 'dmath';
9893: { Unweighted polynomial regression by singular value decomposition }
9894: procedure WSVDPolFit(X, Y, S : TVector;
9895:                         Lb, Ub, Deg : Integer;
9896:                         SVDTol   : Float;
9897:                         B          : TVector;
9898:                         V          : TMatrix); external 'dmath';
9899: { Weighted polynomial regression by singular value decomposition }
9900: procedure RegTest(Y, Ycalc : TVector;
9901:                      LbY, UbY : Integer;
9902:                      V          : TMatrix;
9903:                      LbV, UbV : Integer;
9904:                      var Test : TRegTest); external 'dmath';
9905: { Test of unweighted regression }
9906: procedure WRegTest(Y, Ycalc, S : TVector;
9907:                      LbY, UbY : Integer;
9908:                      V          : TMatrix;
9909:                      LbV, UbV : Integer;
9910:                      var Test : TRegTest); external 'dmath';
9911: { Test of weighted regression }
9912: { -----
9913:   Nonlinear regression
9914:   ----- }
9915: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9916: { Sets the optimization algorithm for nonlinear regression }
9917: function GetOptAlgo : TOptAlgo; external 'dmath';
9918: { Returns the optimization algorithm }
9919: procedure SetMaxParam(N : Byte); external 'dmath';
9920: { Sets the maximum number of regression parameters for nonlinear regression }
9921: function GetMaxParam : Byte; external 'dmath';
9922: { Returns the maximum number of regression parameters for nonlinear regression }
9923: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9924: { Sets the bounds on the I-th regression parameter }
9925: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9926: { Returns the bounds on the I-th regression parameter }
9927: procedure NLFit(RegFunc : TRegFunc;
9928:                     DerivProc : TDervProc;
9929:                     X, Y      : TVector;
9930:                     Lb, Ub    : Integer;
9931:                     MaxIter  : Integer;
9932:                     Tol       : Float;
9933:                     B          : TVector;
9934:                     FirstPar,
9935:                     LastPar   : Integer;
9936:                     V          : TMatrix); external 'dmath';
9937: { Unweighted nonlinear regression }
9938: procedure WNLFit(RegFunc : TRegFunc;
9939:                     DerivProc : TDervProc;
9940:                     X, Y, S   : TVector;
9941:                     Lb, Ub    : Integer;
9942:                     MaxIter  : Integer;
9943:                     Tol       : Float;
9944:                     B          : TVector;
9945:                     FirstPar,
9946:                     LastPar   : Integer;
9947:                     V          : TMatrix); external 'dmath';
9948: { Weighted nonlinear regression }
9949: procedure SetMCFFile(FileName : String); external 'dmath';
9950: { Set file for saving MCMC simulations }
9951: procedure SimFit(RegFunc : TRegFunc;
9952:                     X, Y      : TVector;
9953:                     Lb, Ub    : Integer;
9954:                     B          : TVector;
9955:                     FirstPar,
9956:                     LastPar   : Integer;
9957:                     V          : TMatrix); external 'dmath';
9958: { Simulation of unweighted nonlinear regression by MCMC }
9959: procedure WSimFit(RegFunc : TRegFunc;
9960:                     X, Y, S   : TVector;
9961:                     Lb, Ub    : Integer;
9962:                     B          : TVector;
9963:                     FirstPar,
9964:                     LastPar   : Integer;
9965:                     V          : TMatrix); external 'dmath';
9966: { Simulation of weighted nonlinear regression by MCMC }
9967: { -----

```

```

9968:  Nonlinear regression models
9969:  -----
9970:  procedure FracFit(X, Y      : TVector;
9971:                      Lb, Ub    : Integer;
9972:                      Deg1, Deg2 : Integer;
9973:                      ConsTerm  : Boolean;
9974:                      MaxIter   : Integer;
9975:                      Tol       : Float;
9976:                      B         : TVector;
9977:                      V         : TMatrix); external 'dmath';
9978: { Unweighted fit of rational fraction }
9979:  procedure WFractFit(X, Y, S   : TVector;
9980:                      Lb, Ub    : Integer;
9981:                      Deg1, Deg2 : Integer;
9982:                      ConsTerm  : Boolean;
9983:                      MaxIter   : Integer;
9984:                      Tol       : Float;
9985:                      B         : TVector;
9986:                      V         : TMatrix); external 'dmath';
9987: { Weighted fit of rational fraction }
9988:
9989:  function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9990: { Returns the value of the rational fraction at point X }
9991:  procedure ExpFit(X, Y      : TVector;
9992:                      Lb, Ub, Nexp : Integer;
9993:                      ConsTerm  : Boolean;
9994:                      MaxIter   : Integer;
9995:                      Tol       : Float;
9996:                      B         : TVector;
9997:                      V         : TMatrix); external 'dmath';
9998: { Unweighted fit of sum of exponentials }
9999:  procedure WExpFit(X, Y, S   : TVector;
10000:                      Lb, Ub, Nexp : Integer;
10001:                      ConsTerm  : Boolean;
10002:                      MaxIter   : Integer;
10003:                      Tol       : Float;
10004:                      B         : TVector;
10005:                      V         : TMatrix); external 'dmath';
10006: { Weighted fit of sum of exponentials }
10007:  function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10008: { Returns the value of the regression function at point X }
10009:  procedure IncExpFit(X, Y      : TVector;
10010:                      Lb, Ub    : Integer;
10011:                      ConsTerm  : Boolean;
10012:                      MaxIter   : Integer;
10013:                      Tol       : Float;
10014:                      B         : TVector;
10015:                      V         : TMatrix); external 'dmath';
10016: { Unweighted fit of model of increasing exponential }
10017:  procedure WIIncExpFit(X, Y, S   : TVector;
10018:                      Lb, Ub    : Integer;
10019:                      ConsTerm  : Boolean;
10020:                      MaxIter   : Integer;
10021:                      Tol       : Float;
10022:                      B         : TVector;
10023:                      V         : TMatrix); external 'dmath';
10024: { Weighted fit of increasing exponential }
10025:  function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10026: { Returns the value of the regression function at point X }
10027:  procedure ExpLinFit(X, Y      : TVector;
10028:                      Lb, Ub    : Integer;
10029:                      MaxIter   : Integer;
10030:                      Tol       : Float;
10031:                      B         : TVector;
10032:                      V         : TMatrix); external 'dmath';
10033: { Unweighted fit of the "exponential + linear" model }
10034:  procedure WExpLinFit(X, Y, S : TVector;
10035:                      Lb, Ub    : Integer;
10036:                      MaxIter   : Integer;
10037:                      Tol       : Float;
10038:                      B         : TVector;
10039:                      V         : TMatrix); external 'dmath';
10040: { Weighted fit of the "exponential + linear" model }
10041:
10042:  function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10043: { Returns the value of the regression function at point X }
10044:  procedure MichFit(X, Y      : TVector;
10045:                      Lb, Ub    : Integer;
10046:                      MaxIter   : Integer;
10047:                      Tol       : Float;
10048:                      B         : TVector;
10049:                      V         : TMatrix); external 'dmath';
10050: { Unweighted fit of Michaelis equation }
10051:  procedure WMichFit(X, Y, S : TVector;
10052:                      Lb, Ub    : Integer;
10053:                      MaxIter   : Integer;
10054:                      Tol       : Float;
10055:                      B         : TVector;
10056:                      V         : TMatrix); external 'dmath';

```

```

10057: { Weighted fit of Michaelis equation }
10058: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10059: { Returns the value of the Michaelis equation at point X }
10060: procedure MintFit(X, Y      : TVector;
10061:                      Lb, Ub   : Integer;
10062:                      MintVar : TMintVar;
10063:                      Fit_S0  : Boolean;
10064:                      MaxIter : Integer;
10065:                      Tol     : Float;
10066:                      B       : TVector;
10067:                      V       : TMatrix); external 'dmath';
10068: { Unweighted fit of the integrated Michaelis equation }
10069: procedure WMintFit(X, Y, S : TVector;
10070:                      Lb, Ub   : Integer;
10071:                      MintVar : TMintVar;
10072:                      Fit_S0  : Boolean;
10073:                      MaxIter : Integer;
10074:                      Tol     : Float;
10075:                      B       : TVector;
10076:                      V       : TMatrix); external 'dmath';
10077: { Weighted fit of the integrated Michaelis equation }
10078: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10079: { Returns the value of the integrated Michaelis equation at point X }
10080: procedure HillFit(X, Y      : TVector;
10081:                      Lb, Ub   : Integer;
10082:                      MaxIter : Integer;
10083:                      Tol     : Float;
10084:                      B       : TVector;
10085:                      V       : TMatrix); external 'dmath';
10086: { Unweighted fit of Hill equation }
10087: procedure WHillFit(X, Y, S : TVector;
10088:                      Lb, Ub   : Integer;
10089:                      MaxIter : Integer;
10090:                      Tol     : Float;
10091:                      B       : TVector;
10092:                      V       : TMatrix); external 'dmath';
10093: { Weighted fit of Hill equation }
10094: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10095: { Returns the value of the Hill equation at point X }
10096: procedure LogiFit(X, Y      : TVector;
10097:                      Lb, Ub   : Integer;
10098:                      ConsTerm : Boolean;
10099:                      General  : Boolean;
10100:                     MaxIter : Integer;
10101:                     Tol     : Float;
10102:                     B       : TVector;
10103:                     V       : TMatrix); external 'dmath';
10104: { Unweighted fit of logistic function }
10105: procedure WLogiFit(X, Y, S : TVector;
10106:                      Lb, Ub   : Integer;
10107:                      ConsTerm : Boolean;
10108:                      General  : Boolean;
10109:                      MaxIter : Integer;
10110:                      Tol     : Float;
10111:                      B       : TVector;
10112:                      V       : TMatrix); external 'dmath';
10113: { Weighted fit of logistic function }
10114: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10115: { Returns the value of the logistic function at point X }
10116: procedure PKFit(X, Y      : TVector;
10117:                      Lb, Ub   : Integer;
10118:                      MaxIter : Integer;
10119:                      Tol     : Float;
10120:                      B       : TVector;
10121:                      V       : TMatrix); external 'dmath';
10122: { Unweighted fit of the acid-base titration curve }
10123: procedure WPKFit(X, Y, S : TVector;
10124:                      Lb, Ub   : Integer;
10125:                      MaxIter : Integer;
10126:                      Tol     : Float;
10127:                      B       : TVector;
10128:                      V       : TMatrix); external 'dmath';
10129: { Weighted fit of the acid-base titration curve }
10130: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10131: { Returns the value of the acid-base titration function at point X }
10132: procedure PowFit(X, Y      : TVector;
10133:                      Lb, Ub   : Integer;
10134:                      MaxIter : Integer;
10135:                      Tol     : Float;
10136:                      B       : TVector;
10137:                      V       : TMatrix); external 'dmath';
10138: { Unweighted fit of power function }
10139: procedure WPowFit(X, Y, S : TVector;
10140:                      Lb, Ub   : Integer;
10141:                      MaxIter : Integer;
10142:                      Tol     : Float;
10143:                      B       : TVector;
10144:                      V       : TMatrix); external 'dmath';
10145: { Weighted fit of power function }

```

```

10146:
10147: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10148: { Returns the value of the power function at point X }
10149: procedure GammaFit(X, Y : TVector;
10150:           Lb, Ub : Integer;
10151:           MaxIter : Integer;
10152:           Tol : Float;
10153:           B : TVector;
10154:           V : TMatrix); external 'dmath';
10155: { Unweighted fit of gamma distribution function }
10156: procedure WGammaFit(X, Y, S : TVector;
10157:           Lb, Ub : Integer;
10158:           MaxIter : Integer;
10159:           Tol : Float;
10160:           B : TVector;
10161:           V : TMatrix); external 'dmath';
10162: { Weighted fit of gamma distribution function }
10163: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10164: { Returns the value of the gamma distribution function at point X }
10165: { -----
10166:   Principal component analysis
10167: ----- }
10168: procedure VecMean(X : TMatrix;
10169:           Lb, Ub, Nvar : Integer;
10170:           M : TVector); external 'dmath';
10171: { Computes the mean vector M from matrix X }
10172: procedure VecSD(X : TMatrix;
10173:           Lb, Ub, Nvar : Integer;
10174:           M, S : TVector); external 'dmath';
10175: { Computes the vector of standard deviations S from matrix X }
10176: procedure MatVarCov(X : TMatrix;
10177:           Lb, Ub, Nvar : Integer;
10178:           M : TVector;
10179:           V : TMatrix); external 'dmath';
10180: { Computes the variance-covariance matrix V from matrix X }
10181: procedure MatCorrel(V : TMatrix;
10182:           Nvar : Integer;
10183:           R : TMatrix); external 'dmath';
10184: { Computes the correlation matrix R from the var-cov matrix V }
10185: procedure PCA(R : TMatrix;
10186:           Nvar : Integer;
10187:           Lambda : TVector;
10188:           C, Rc : TMatrix); external 'dmath';
10189: { Performs a principal component analysis of the correlation matrix R }
10190: procedure ScaleVar(X : TMatrix;
10191:           Lb, Ub, Nvar : Integer;
10192:           M, S : TVector;
10193:           Z : TMatrix); external 'dmath';
10194: { Scales a set of variables by subtracting means and dividing by SD's }
10195: procedure PrinFac(Z : TMatrix;
10196:           Lb, Ub, Nvar : Integer;
10197:           C, F : TMatrix); external 'dmath';
10198: { Computes principal factors
10199: ----- }
10200: Strings
10201: ----- }

10202: function LTrim(S : String) : String; external 'dmath';
10203: { Removes leading blanks }
10204: function RTrim(S : String) : String; external 'dmath';
10205: { Removes trailing blanks }
10206: function Trim(S : String) : String; external 'dmath';
10207: { Removes leading and trailing blanks }
10208: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10209: { Returns a string made of character C repeated N times }
10210: function RFill(S : String; L : Byte) : String; external 'dmath';
10211: { Completes string S with trailing blanks for a total length L }
10212: function LFill(S : String; L : Byte) : String; external 'dmath';
10213: { Completes string S with leading blanks for a total length L }
10214: function CFill(S : String; L : Byte) : String; external 'dmath';
10215: { Centers string S on a total length L }
10216: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10217: { Replaces in string S all the occurrences of C1 by C2 }
10218: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10219: { Extracts a field from a string }
10220: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10221: { Parses a string into its constitutive fields }
10222: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10223: { Sets the numeric format }
10224: function FloatStr(X : Float) : String; external 'dmath';
10225: { Converts a real to a string according to the numeric format }
10226: function IntStr(N : LongInt) : String; external 'dmath';
10227: { Converts an integer to a string }
10228: function CompStr(Z : Complex) : String; external 'dmath';
10229: { Converts a complex number to a string }
10230: {$IFDEF DELPHI}
10231: function StrDec(S : String) : String; external 'dmath';
10232: { Set decimal separator to the symbol defined in SysUtils }
10233: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10234: { Test if a string represents a number and returns it in X }

```

```

10235: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10236: { Reads a floating point number from an Edit control }
10237: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10238: { Writes a floating point number in a text file }
10239: {$ENDIF}
10240: {
10241:   ----- BGI / Delphi graphics -----
10242: }
10243: function InitGraphics
10244: {$IFDEF DELPHI}
10245: (Width, Height : Integer) : Boolean;
10246: {$ELSE}
10247: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10248: { Enters graphic mode }
10249: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10250: X1, X2, Y1, Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10251: { Sets the graphic window }
10252: procedure SetOxScale(Scale : TScale;
10253: Oxmin, OxMax, OxStep : Float); external 'dmath';
10254: { Sets the scale on the Ox axis }
10255: procedure SetOyScale(Scale : TScale;
10256: OyMin, OyMax, OyStep : Float); external 'dmath';
10257: { Sets the scale on the Oy axis }
10258: procedure GetOxScale(var Scale : TScale;
10259: var OxMin, OxMax, OxStep : Float); external 'dmath';
10260: { Returns the scale on the Ox axis }
10261: procedure GetOyScale(var Scale : TScale;
10262: var OyMin, OyMax, OyStep : Float); external 'dmath';
10263: { Returns the scale on the Oy axis }
10264: procedure SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10265: procedure SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10266: procedure SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10267: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10268: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10269: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10270: {$IFNDEF DELPHI}
10271: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10272: { Sets the font for the main graph title }
10273: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10274: { Sets the font for the Ox axis (title and labels) }
10275: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10276: { Sets the font for the Oy axis (title and labels) }
10277: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10278: { Sets the font for the legends }
10279: procedure SetClipping(Clip : Boolean); external 'dmath';
10280: { Determines whether drawings are clipped at the current viewport
10281:   boundaries, according to the value of the Boolean parameter Clip }
10282: {$ENDIF}
10283: procedure PlotOxAxis({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10284: { Plots the horizontal axis }
10285: procedure PlotOyAxis({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10286: { Plots the vertical axis }
10287: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10288: { Plots a grid on the graph }
10289: procedure WriteGraphTitle({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10290: { Writes the title of the graph }
10291: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10292: { Sets the maximum number of curves and re-initializes their parameters }
10293: procedure SetPointParam
10294: {$IFDEF DELPHI}
10295: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10296: {$ELSE}
10297: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10298: { Sets the point parameters for curve # CurvIndex }
10299: procedure SetLineParam
10300: {$IFDEF DELPHI}
10301: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10302: {$ELSE}
10303: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10304: { Sets the line parameters for curve # CurvIndex }
10305: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10306: { Sets the legend for curve # CurvIndex }
10307: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10308: { Sets the step for curve # CurvIndex }
10309: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10310: procedure GetPointParam
10311: {$IFDEF DELPHI}
10312: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10313: {$ELSE}
10314: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10315: { Returns the point parameters for curve # CurvIndex }
10316: procedure GetLineParam
10317: {$IFDEF DELPHI}
10318: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10319: {$ELSE}
10320: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10321: { Returns the line parameters for curve # CurvIndex }
10322: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10323: { Returns the legend for curve # CurvIndex }

```

```

10324: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10325: { Returns the step for curve # CurvIndex }
10326: {$IFDEF DELPHI}
10327: procedure PlotPoint(Canvas      : TCanvas;
10328:                      X, Y       : Float; CurvIndex : Integer); external 'dmath';
10329: {$ELSE}
10330: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10331: {$ENDIF}
10332: { Plots a point on the screen }
10333: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10334:                      X, Y           : TVector;
10335:                      Lb, Ub, CurvIndex : Integer); external 'dmath';
10336: { Plots a curve }
10337: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10338:                      X, Y, S          : TVector;
10339:                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10340: { Plots a curve with error bars }
10341: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10342:                      Func        : TFunc;
10343:                      Xmin, Xmax   : Float;
10344:                      {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10345:                      CurvIndex    : Integer); external 'dmath';
10346: { Plots a function }
10347: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10348:                      NCurv      : Integer;
10349:                      ShowPoints, ShowLines : Boolean); external 'dmath';
10350: { Writes the legends for the plotted curves }
10351: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10352:                      Nx, Ny, Nc   : Integer;
10353:                      X, Y, Z     : TVector;
10354:                      F            : TMatrix); external 'dmath';
10355: { Contour plot }
10356: function Xpixel(X : Float):Integer; external 'dmath'; { Converts user abscissa X to screen coordinate }
10357: function Ypixel(Y : Float):Integer; external 'dmath'; { Converts user ordinate Y to screen coordinate }
10358: function Xuser(X : Integer):Float; external 'dmath'; { Converts screen coordinate X to user abscissa }
10359: function Yuser(Y : Integer):Float; external 'dmath'; { Converts screen coordinate Y to user ordinate }
10360: {$IFDEF DELPHI}
10361: procedure LeaveGraphics; external 'dmath';
10362: { Quits graphic mode }
10363: {$ENDIF}
10364: { -----
10365:   LaTeX graphics
10366:   ----- }
10367: function TeX_InitGraphics(FileName      : String; PgWidth, PgHeight : Integer;
10368:                           Header        : Boolean); external 'dmath';
10369: { Initializes the LaTeX file }
10370: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10371: { Sets the graphic window }
10372: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10373: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10374: { Sets the scale on the Ox axis }
10375: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10376: { Sets the scale on the Oy axis }
10377: procedure TeX_SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10378: procedure TeX_SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10379: procedure TeX_SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10380: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10381: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10382: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10383: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10384: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10385: { Sets the maximum number of curves and re-initializes their parameters }
10386: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10387: { Sets the point parameters for curve # CurvIndex }
10388: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10389:                           Width : Float; Smooth : Boolean); external 'dmath';
10390: { Sets the line parameters for curve # CurvIndex }
10391: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10392: { Sets the legend for curve # CurvIndex }
10393: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10394: { Sets the step for curve # CurvIndex }
10395: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10396: { Plots a curve }
10397: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10398:                                         Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10399: { Plots a curve with error bars }
10400: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10401:                           Npt : Integer; CurvIndex : Integer); external 'dmath';
10402: { Plots a function }
10403: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10404: { Writes the legends for the plotted curves }
10405: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10406: { Contour plot }
10407: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10408: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10409:
10410: //*****unit uPSI_SynPdf;
10411: Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10412: Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate

```

```

10413: Function _PfdDateToDateTime( const AText : TPdfDate ) : TDateTime
10414: Function PdfRect( Left, Top, Right, Bottom : Single ) : TPdfRect;
10415: Function PdfRect1( const Box : TPdfBox ) : TPdfRect;
10416: Function PdfBox( Left, Top, Width, Height : Single ) : TPdfBox
10417: //Function _GetCharCount( Text : PAnsiChar ) : integer
10418: //Procedure L2R( W : PWideChar; L : integer)
10419: Function PdfCoord( MM : single ) : integer
10420: Function CurrentPrinterPageSize : TPDFPaperSize
10421: Function CurrentPrinterRes : TPoint
10422: Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8 )
10423: Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer )
10424: Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect )
10425: Const ('Usp10', 'String 'usp10.dll
10426: AddTypes('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10427: 'fcharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10428: TScriptState_set', 'set of TScriptState_enum
10429: //*****unit uPSI_TlHelp32;
10430:
10431: procedure SIRегистre_PMrand(CL: TPPascalCompiler); //ParkMiller
10432: begin
10433: Procedure PMrandomize( I : word)
10434: Function PMrandom : longint
10435: Function Rrand : extended
10436: Function Irand( N : word) : word
10437: Function Brand( P : extended) : boolean
10438: Function Nrand : extended
10439: end;
10440:
10441: procedure SIRегистre_Spring_Cryptography_Utils(CL: TPPascalCompiler);
10442: begin
10443: Function Endian( x : LongWord) : LongWord
10444: Function Endian64( x : Int64) : Int64
10445: Function spRol( x : LongWord; y : Byte) : LongWord
10446: Function spRor( x : LongWord; y : Byte) : LongWord
10447: Function Ror64( x : Int64; y : Byte) : Int64
10448: end;
10449:
10450: procedure SIRегистre_MapReader(CL: TPPascalCompiler);
10451: begin
10452: Procedure ClearModules
10453: Procedure ReadMapfile( Fname : string)
10454: Function AddressInfo( Address : dword) : string
10455: end;
10456:
10457: procedure SIRегистre_LibTar(CL: TPPascalCompiler);
10458: begin
10459: TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10460: +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10461: +'teByOther, tpExecuteByOther )
10462: TTarPermissions', 'set of TTarPermission
10463: TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10464: +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10465: TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10466: TTarModes', 'set of TTarMode
10467: TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDA'
10468: +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10469: +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING; '
10470: +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10471: +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10472: SIRегистre_TTarArchive(CL);
10473: SIRегистre_TTarWriter(CL);
10474: Function PermissionString( Permissions : TTarPermissions ) : STRING
10475: Function ConvertFilename( Filename : STRING ) : STRING
10476: Function FileTimeGMT( FileName : STRING ) : TDateTime;
10477: Function FileTimeGMT1( SearchRec : TSearchRec ) : TDateTime;
10478: Procedure ClearDirRec( var DirRec : TTarDirRec )
10479: end;
10480:
10481:
10482: //*****unit uPSI_TlHelp32;
10483: procedure SIRегистre_TlHelp32(CL: TPPascalCompiler);
10484: begin
10485: Const ('MAX_MODULE_NAME32', 'LongInt'( 255 );
10486: Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD ) : THandle
10487: Const ('TH32CS_SNAPHEAPLIST', 'LongWord( $00000001 );
10488: Const ('TH32CS_SNAPPROCESS', 'LongWord').SetUInt( $00000002 );
10489: Const ('TH32CS_SNAPTHREAD', 'LongWord').SetUInt( $00000004 );
10490: Const ('TH32CS_SNAPMODULE', 'LongWord').SetUInt( $00000008 );
10491: Const ('TH32CS_INHERIT', 'LongWord').SetUInt( $80000000 );
10492: tagHEAPLIST32', 'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10493: AddTypes('HEAPLIST32', 'tagHEAPLIST32
10494: AddTypes('THeapList32', 'tagHEAPLIST32
10495: Const ('HF32_DEFAULT', 'LongInt'( 1 );
10496: Const ('HF32_SHARED', 'LongInt'( 2 );
10497: Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10498: Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10499: AddTypes('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAdr'
10500: +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10501: +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end

```

```

10502: AddTypes('HEAPENTRY32', 'tagHEAPENTRY32'
10503: AddTypes('THeapEntry32', 'tagHEAPENTRY32'
10504: Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10505: Const('LF32_FREE','LongWord').SetUInt( $00000002);
10506: Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10507: Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD ) : BOOL
10508: Function Heap32Next( var lphe : THeapEntry32 ) : BOOL
10509: DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10510: AddTypes('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10511: +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10512: +'aPri : Longint; dwFlags : DWORD; end
10513: AddTypes('TTHREADENTRY32', 'tagTHREADENTRY32'
10514: AddTypes('TThreadEntry32', 'tagTHREADENTRY32'
10515: Function Thread32First( hSnapshot : THandle; var lppe : TThreadEntry32 ) : BOOL
10516: Function Thread32Next( hSnapshot : THandle; var lppe : TThreadEntry32 ) : BOOL
10517: end;
10518: Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10519: Const('EW_REBOOTSYSTEM','LongWord( $0043);
10520: Const('EW_EXITANDEXECAPP','LongWord( $0044);
10521: Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ) );
10522: Const('EWX_LOGOFF','LongInt'( 0);
10523: Const('EWX_SHUTDOWN','LongInt'( 1);
10524: Const('EWX_REBOOT','LongInt'( 2);
10525: Const('EWX_FORCE','LongInt'( 4);
10526: Const('EWX_POWEROFF','LongInt'( 8);
10527: Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10528: Function GET_APPCOMMAND_LPARAM( const lParam : LongInt ) : Shortint
10529: Function GET_DEVICE_LPARAM( const lParam : LongInt ) : Word
10530: Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt ) : Word
10531: Function GET_FLAGS_LPARAM( const lParam : LongInt ) : Word
10532: Function GET_KEYSTATE_LPARAM( const lParam : LongInt ) : Word
10533: Function GetWindowWord( hWnd : HWND; nIndex : Integer ) : Word
10534: Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word
10535: Function GetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
10536: Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
10537: Function GetClassWord( hWnd : HWND; nIndex : Integer ) : Word
10538: Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word
10539: Function GetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
10540: Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
10541: Function GetDesktopWindow : HWND
10542: Function GetParent( hWnd : HWND ) : HWND
10543: Function SetParent( hWndChild, hWndNewParent : HWND ) : HWND
10544: Function GetTopWindow( hWnd : HWND ) : HWND
10545: Function GetNextWindow( hWnd : HWND; uCmd : UINT ) : HWND
10546: Function GetWindow( hWnd : HWND; uCmd : UINT ) : HWND
10547: //Delphi DFM
10548: Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10549: Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string ) : integer
10550: procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10551: function GetHighlightersFilter(AHighlighters: TStringList): string;
10552: function GetHighlighterFromfileExt(AHighlighters: TStringList; Extension: string): TSynCustomHighlighter;
10553: Function ShowOwnedPopups(hWnd : HWND; fShow : BOOL) : BOOL
10554: Function OpenIcon( hWnd : HWND ) : BOOL
10555: Function CloseWindow( hWnd : HWND ) : BOOL
10556: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL ) : BOOL
10557: Function SetWindowPos(hWnd:HWND;hWndInsertAfter:HWND; X,Y,cx,cy:Integer;uFlags:UINT) : BOOL
10558: Function IsWindowVisible( hWnd : HWND ) : BOOL
10559: Function IsIconic( hWnd : HWND ) : BOOL
10560: Function AnyPopup : BOOL
10561: Function BringWindowToTop( hWnd : HWND ) : BOOL
10562: Function IsZoomed( hWnd : HWND ) : BOOL
10563: Function IsWindow( hWnd : HWND ) : BOOL
10564: Function IsMenu( hMenu : HMENU ) : BOOL
10565: Function IsChild( hWndParent, hWnd : HWND ) : BOOL
10566: Function DestroyWindow( hWnd : HWND ) : BOOL
10567: Function ShowWindow( hWnd : HWND; nCmdShow : Integer ) : BOOL
10568: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD ) : BOOL
10569: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer ) : BOOL
10570: Function FlashWindow( hWnd : HWND; bInvert : BOOL ) : BOOL
10571: Function IsWindowUnicode( hWnd : HWND ) : BOOL
10572: Function EnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL
10573: Function IsWindowEnabled( hWnd : HWND ) : BOOL
10574:
10575: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10576: begin
10577:   const ('ShowSetupDialogOptLong','String '--setup
10578: PrimaryConfPathOptLong','String '--primary-config-path=
10579: PrimaryConfPathOptShort','String '--pcp=
10580: SecondaryConfPathOptLong','String '--secondary-config-path=
10581: SecondaryConfPathOptShort','String '--scp=
10582: NoSplashScreenOptLong','String '--no-splash-screen
10583: NoSplashScreenOptShort','String '--nsc
10584: StartedByStartLazarusOpt','String '--started-by-startlazarus
10585: SkipLastProjectOpt','String '--skip-last-project
10586: DebugLogOpt','String '--debug-log=
10587: DebugLogOptEnable','String '--debug-enable=
10588: LanguageOpt','String '--language=
10589: LazarusDirOpt','String '--lazarusdir=
10590: Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:bool);

```

```

10591: Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10592: Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10593: Function IsHelpRequested : Boolean
10594: Function IsVersionRequested : boolean
10595: Function GetLanguageSpecified : string
10596: Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10597: Function ParamIsOptionPlusValue(ParamIndex:int;const Option:string;out AValue:string):bool;
10598: Procedure ParseNoGuiCmdLineParams
10599: Function ExtractCmdLineFilenames : TStrings
10600: end;
10601:
10602:
10603: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10604: begin
10605:   Function CompareFilenames( const Filename1,Filename2 : string) : integer
10606:   Function CompareFilenamesIgnoreCase( const Ffilename1,Filename2 : string) : integer
10607:   Function CompareFileExt( const Ffilename, Ext : string; CaseSensitive : boolean) : integer;
10608:   Function CompareFileExt1( const Ffilename, Ext : string) : integer;
10609:   Function CompareFilenameStarts( const Ffilename, Ffilename2 : string) : integer
10610:   Function CompareFilenames(Ffilename:PChar;Len1:integer; Ffilename2:PChar;Len2:integer):integer
10611:   Function CompareFilenamesP( Ffilename, Ffilename2 : PChar; IgnoreCase : boolean) : integer
10612:   Function DirPathExists( DirectoryName : string) : boolean
10613:   Function DirectoryIsWritable( const DirectoryName : string) : boolean
10614:   Function ExtractFileNameOnly( const AFilename : string) : string
10615:   Function FfilenameIsAbsolute( const TheFfilename : string) : boolean
10616:   Function FfilenameIsWinAbsolute( const TheFfilename : string) : boolean
10617:   Function FfilenameIsUnixAbsolute( const TheFfilename : string) : boolean
10618:   Function ForceDirectory( DirectoryName : string) : boolean
10619:   Procedure CheckIfFileIsExecutable( const AFilename : string)
10620:   Procedure CheckIfFileIsSymlink( const AFilename : string)
10621:   Function FileIsText( const AFilename : string) : boolean
10622:   Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10623:   Function FfilenameIsTrimmed( const TheFfilename : string) : boolean
10624:   Function FfilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10625:   Function TrimFfilename( const AFilename : string) : string
10626:   Function ResolveDots( const AFilename : string) : string
10627:   Procedure ForcePathDelims( var FileName : string)
10628:   Function GetForcedPathDelims( const FileName : string) : String
10629:   Function CleanAndExpandfilename( const Ffilename : string) : string
10630:   Function CleanAndExpandDirectory( const Ffilename : string) : string
10631:   Function TrimAndExpandfilename( const Ffilename : string; const BaseDir : string) : string
10632:   Function TrimAndExpandDirectory( const Ffilename : string; const BaseDir : string) : string
10633:   Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10634:   Function CreateRelativePath( const Ffilename,BaseDirectory:string; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder: Boolean) : string
10635:   Function FileIsInPath( const Ffilename, Path : string) : boolean
10636:   Function AppendPathDelim( const Path : string) : string
10637:   Function ChompPathDelim( const Path : string) : string
10638:   Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10639:   Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10640:   Function MinimizeSearchPath( const SearchPath : string) : string
10641:   Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int) : PChar;
10642: (*Function FileExistsUTF8( const Ffilename : string) : boolean
10643: Function FileAgeUTF8( const Ffilename : string) : Longint
10644: Function DirectoryExistsUTF8( const Directory : string) : Boolean
10645: Function ExpandFileNameUTF8( const Ffilename : string; BaseDir : string) : string
10646: Function FindFirstUTF8(const Path:istring; Attr: Longint; out Rslt : TSearchRec) : Longint
10647: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10648: Procedure FindCloseUTF8( var F : TSearchrec)
10649: Function FileSetDateUTF8( const Ffilename : String; Age : Longint) : Longint
10650: Function FileGetAttrUTF8( const Ffilename : String) : Longint
10651: Function FileSetAttrUTF8( const Ffilename : String; Attr : longint) : Longint
10652: Function DeleteFileUTF8( const Ffilename : String) : Boolean
10653: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10654: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10655: Function FileIsReadOnlyUTF8( const Ffilename : String) : Boolean
10656: Function GetCurrentDirUTF8 : String
10657: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10658: Function CreateDirUTF8( const NewDir : String) : Boolean
10659: Function RemoveDirUTF8( const Dir : String) : Boolean
10660: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10661: Function FileOpenUTF8( const Ffilename : string; Mode : Integer) : THandle
10662: Function FileCreateUTF8( const Ffilename : string) : THandle;
10663: Function FileCreateUTF81( const Ffilename : string; Rights : Cardinal) : THandle;
10664: Function FileCreateUtf82( const Ffilename : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10665: Function FileSizeUtf8( const Ffilename : string) : int64
10666: Function GetFileDescription( const AFilename : string) : string
10667: Function GetAppConfigDirUTF8( Global: Boolean; Create : boolean) : string
10668: Function GetAppConfigFileUTF8( Global: Boolean; SubDir: boolean; CreateDir: boolean) : string
10669: Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10670: Function IsUNCPath( const Path : String) : Boolean
10671: Function ExtractUNCVolume( const Path : String) : String
10672: Function ExtractFileRoot( Ffilename : String) : String
10673: Function GetDarwinSystemFilename( Ffilename : string) : string
10674: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10675: Function StrToCmdLineParam( const Param : string) : string
10676: Function MergeCmdLineParams( ParamList : TStrings) : string
10677: Procedure InvalidateFileStateCache( const Ffilename : string)

```

```

10678: Function FindAllFiles( const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean) : TStringList;
10679: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10680: Function FindAllDocs( const Root, extmask: string): TStringlist;
10681: Function ReadfileToString( const Filename : string) : string
10682: procedure Incl(var X: longint; N: Longint);
10683:
10684: type
10685:   TCopyFileFlag = ( cffOverwriteFile,
10686:                      cffCreateDestDirectory, cffPreserveTime );
10687:   TCopyFileFlags = set of TCopyFileFlag,*)
10688:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10689:   TCopyFileFlags', 'set of TCopyFileFlag
10690:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10691: end;
10692:
10693: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10694: begin
10695:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10696:   SIRegister_TMask(CL);
10697:   SIRegister_TParseStringList(CL);
10698:   SIRegister_TMaskList(CL);
10699:   Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10700:   Function MatchesWindowsMask(const FileName,Mask:String;const CaseSensitive : Boolean) : Bool;
10701:   Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10702:   Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10703: end;
10704:
10705: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10706: begin
10707:   //PShellHookInfo', '^TShellHookInfo // will not work
10708:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10709:   SHELLHOOKINFO', 'TShellHookInfo
10710:   LPSHELLHOOKINFO', 'PShellHookInfo
10711:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10712:   SIRegister_TJvShellHook(CL);
10713:   Function InitJvShellHooks : Boolean
10714:   Procedure UnInitJvShellHooks
10715: end;
10716:
10717: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10718: begin
10719:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10720:   '+', dcHasSelSel, dcWantTab, dcNative )
10721:   TDlgCodes', 'set of TDlgCode
10722:   'dcWantMessage', ' dcWantAllKeys);
10723:   SIRegister_IJvExControl(CL);
10724:   SIRegister_IJvDenySubClassing(CL);
10725:   SIRegister_TStructPtrMessage(CL);
10726:   Procedure SetDotNetBarColors( FocusedColor, UnfocusedColor : TColor)
10727:   Procedure DrawDotNetBar( Control : TWinControl; AColor : TColor; InControl : Boolean);
10728:   Procedure DrawDotNetBar(DC: HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10729:   Procedure HandleDotNetBarHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10730:   Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10731:   Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10732:   Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10733:   Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10734:   Function GetFocusedControl( AControl : TControl) : TWinControl
10735:   Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10736:   Function DlgCodesToDlcg( Value : TDlgCodes) : Longint
10737:   Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10738:   Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10739:   SIRegister_TJvExControl(CL);
10740:   SIRegister_TJvExWinControl(CL);
10741:   SIRegister_TJvExCustomControl(CL);
10742:   SIRegister_TJvExGraphicControl(CL);
10743:   SIRegister_TJvExHintWindow(CL);
10744:   SIRegister_TJvExPubGraphicControl(CL);
10745: end;
10746:
10747: (*-----*)
10748: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10749: begin
10750:   Procedure EncodeStream( Input, Output : TStream)
10751:   Procedure DecodeStream( Input, Output : TStream)
10752:   Function EncodeString1( const Input : string) : string
10753:   Function DecodeString1( const Input : string) : string
10754: end;
10755:
10756: (*-----*)
10757: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10758: begin
10759:   SIRegister_TWebAppRegInfo(CL);
10760:   SIRegister_TWebAppRegList(CL);
10761:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10762:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10763:   Procedure UnregisterWebApp( const AProgID : string)
10764:   Function FindRegisteredWebApp( const AProgID : string) : string
10765:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10766:   'sUDPPort','String 'UDPPort

```

```

10767: end;
10768:
10769: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10770: begin
10771: // TStringDynArray', 'array of string
10772: Function GetEnvVarValue( const VarName : string ) : string
10773: Function SetEnvVarValue( const VarName, VarValue : string ) : Integer
10774: Function DeleteEnvVar( const VarName : string ) : Integer
10775: Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
BufSize:Int):Int;
10776: Function ExpandEnvVars( const Str : string ) : string
10777: Function GetAllEnvVars( const Vars : TStrings ) : Integer
10778: Procedure GetAllEnvVarNames( const Names : TStrings );
10779: Function GetAllEnvVarNames1 : TStringDynArray;
10780: Function EnvBlockSize : Integer
10781: TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10782: SIRegister_TPJEnvVarsEnumerator(CL);
10783: SIRegister_TPJEnvVars(CL);
10784: FindClass('TOBJECT'), 'EPJEnvVars
10785: FindClass('TOBJECT'), 'EPJEnvVars
10786: //Procedure Register
10787: end;
10788:
10789: (*-----*)
10790: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10791: begin
10792: 'cOneSecInMS','LongInt'( 1000 );
10793: // 'cDefTimeSlice','LongInt'( 50 );
10794: // 'cDefMaxExecTime',' cOneMinInMS';
10795: 'cAppErrorMask','LongInt'( 1 shl 29 );
10796: Function IsApplicationError( const ErrCode : LongWord ) : Boolean
10797: TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10798: TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10799: Function MakeConsoleColors(const AForeground, ABackground: TPJConsoleColor):TPJConsoleColors;
10800: Function MakeConsoleColors( const AForeground, ABackground : TColor ) : TPJConsoleColors;
10801: Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor ) : TPJConsoleColors;
10802: Function MakeSize( const ACX, ACY : LongInt ) : TSize
10803: SIRegister_TPJCustomConsoleApp(CL);
10804: SIRegister_TPJConsoleApp(CL);
10805: end;
10806:
10807: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10808: begin
10809: INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff );
10810: t_encoding', '( uencode, base64, mime )
10811: Function internet_date( date : TDateTime ) : string
10812: Function lookup_hostname( const hostname : string ) : longint
10813: Function my_hostname : string
10814: Function my_ip_address : longint
10815: Function ip2string( ip_address : longint ) : string
10816: Function resolve_hostname( ip : longint ) : string
10817: Function address_from( const s : string; count : integer ) : string
10818: Function encode_base64( data : TStream ) : TStringList
10819: Function decode_base64( source : TStringList ) : TMemoryStream
10820: Function posn( const s, t : string; count : integer ) : integer
10821: Function poscn( c : char; const s : string; n : integer ) : integer
10822: Function filename_of( const s : string ) : string
10823: //Function trim( const s : string ) : string
10824: //Procedure setlength( var s : string; l : byte)
10825: Function TimeZoneBias : longint
10826: Function eight2seven_quoteprint( const s : string ) : string
10827: Function eight2seven_german( const s : string ) : string
10828: Function seven2eight_quoteprint( const s : string ) : string end;
10829: type in_addr', 'record s_bytes : array[1..4] of byte; end;
10830: Function socketerror : cint
10831: Function fsocket( domain : cint; xtype : cint; protocol : cint ) : cint
10832: Function fprevc( s : cint; buf : __pointer; len : size_t; flags : cint ) : ssize_t
10833: Function fpssend( s : cint; msg : __pointer; len : size_t; flags : cint ) : ssize_t
10834: //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen ) : cint
10835: Function fplistens( s : cint; backlog : cint ) : cint
10836: //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint ) : cint
10837: //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen ) : cint
10838: //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen ) : cint
10839: Function NetAddrToStr( Entry : in_addr ) : String
10840: Function HostAddrToStr( Entry : in_addr ) : String
10841: Function StrToHostAddr( IP : String ) : in_addr
10842: Function StrToNetAddr( IP : String ) : in_addr
10843: SOL_SOCKET','LongWord').SetUInt( $ffff );
10844: cint8', 'shortint
10845: cuint8', 'byte
10846: cchar', 'cint8
10847: cschar', 'cint8
10848: uchar', 'cuint8
10849: cint16', 'smallint
10850: cuint16', 'word
10851: cshort', 'cint16
10852: csshort', 'cint16
10853: cushort', 'cuint16
10854: cint32', 'longint

```

```

10855:    cuint32', 'longword
10856:    cint', 'cint32
10857:    csint', 'cint32
10858:    cuint', 'cuint32
10859:    csigned', 'cint
10860:    cunsigned', 'cuint
10861:    cint64', 'int64
10862:    clonglong', 'cint64
10863:    cslonglong', 'cint64
10864:    cbool', 'longbool
10865:    cfloat', 'single
10866:    cdouble', 'double
10867:    clongdouble', 'extended
10868:
10869: procedure SIRегистер_uLkJSON(CL: TPSPPascalCompiler);
10870: begin
10871:   TlkJSONTypes', '(jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10872:   SIRегистер_TlkJSONdotnetclass(CL);
10873:   SIRегистер_TlkJSONbase(CL);
10874:   SIRегистер_TlkJSONnumber(CL);
10875:   SIRегистер_TlkJSONstring(CL);
10876:   SIRегистер_TlkJSONboolean(CL);
10877:   SIRегистер_TlkJSONnull(CL);
10878:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10879:     +'se; data : TObject; var Continue : Boolean)
10880:   SIRегистер_TlkJSONcustomlist(CL);
10881:   SIRегистер_TlkJSONlist(CL);
10882:   SIRегистер_TlkJSONobjectmethod(CL);
10883:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10884:   TlkHashFunction', 'Function ( const ws : WideString ) : cardinal
10885:   SIRегистер_TlkHashTable(CL);
10886:   SIRегистер_TlkBalTree(CL);
10887:   SIRегистер_TlkJSONobject(CL);
10888:   SIRегистер_TlkJSON(CL);
10889:   SIRегистер_TlkJSONstreamed(CL);
10890:   Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer) : string
10891: end;
10892:
10893: procedure SIRегистер_ZSysUtils(CL: TPSPPascalCompiler);
10894: begin
10895:   TZListSortCompare', 'Function (Item1, Item2 : TObject) : Integer
10896:   SIRегистер_TZSortedlist(CL);
10897:   Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10898:   Function zLastDelimiter( const Delimiters, Str : string) : Integer
10899:   //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10900:   //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10901:   Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10902:   Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10903:   Function EndsWith( const Str, SubStr : WideString) : Boolean;
10904:   Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10905:   Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10906:   Function SQLStrToFloatDef1( Str : string; Def : Extended) : Extended;
10907:   Function SQLStrToFloat( const Str : AnsiString) : Extended
10908:   //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10909:   //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10910:   Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10911:   Function StrToBoolEx( Str : string) : Boolean
10912:   Function BoolToStrEx( Bool : Boolean) : string
10913:   Function IsIpAddr( const Str : string) : Boolean //IsIP()
10914:   Function zSplitString( const Str, Delimiters : string) : TStrings
10915:   Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10916:   Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10917:   Function ComposeString( List : TStrings; const Delimiter : string) : string
10918:   Function FloatToSQLStr( Value : Extended) : string
10919:   Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10920:   Function SplitStringEx( const Str, Delimiter : string) : TStrings
10921:   Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10922:   Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10923:   Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10924:   Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10925:   Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10926:   Function StrToBytes3( const Value : WideString) : TByteDynArray;
10927:   Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10928:   Function BytesToVar( const Value : TByteDynArray) : Variant
10929:   Function VarToBytes( const Value : Variant) : TByteDynArray
10930:   Function AnsiSQLDateToDate( const Value : string) : TDateTime
10931:   Function TimestampStrToDate( const Value : string) : TDateTime
10932:   Function DateToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10933:   Function EncodeCString( const Value : string) : string
10934:   Function DecodeCString( const Value : string) : string
10935:   Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10936:   Function MemPas( Buffer : PChar; Length : LongInt) : string
10937:   Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
SubVersion:Int);
10938:   Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Int;
10939:   Function FormatsSQLVersion( const SQLVersion : Integer) : String
10940:   Function ZStrToFloat( Value : AnsiChar) : Extended;
10941:   Function ZStrToFloat1( Value : AnsiString) : Extended;

```

```

10942: Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10943: Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10944: Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10945: Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10946: Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10947: Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10948: Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10949: end;
10950:
10951: unit uPSI_ZEncoding;
10952: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10953: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10954: Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10955: Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10956: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10957: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10958: Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10959: Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10960: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10961: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10962: Function ZConvertRawToString( const Src:RawByteString; const RawCP, StringCP : Word):String
10963: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10964: Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10965: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10966: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10967: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP : Word):UTF8String
10968: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10969: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10970: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10971: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10972: Function ZConvertUnicodeToString_CPUTF8( const Src:WideString; const StringCP : Word): String
10973: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10974: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10975: Function ZConvertStringToUnicodeWithAutoEncode( const Src : String; const StringCP:Word):WideString
10976: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10977: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10978: Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10979: Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10980: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10981: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10982: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10983: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10984: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10985: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10986: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10987: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10988: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10989: Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10990: Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10991: Function ZDefaultSystemCodePage : Word
10992: Function ZCompatibleCodePages( const CPL, CP2 : Word) : Boolean
10993: function MPing(const AHost: string;const ATimes:integer; out AvgMS:Double):Boolean;
10994:
10995: procedure SIRegister_BoldComUtils(CL: TPSPPascalCompiler);
10996: begin
10997:   ('RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10998:   ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10999:   ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
11000:   ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
11001:   ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
11002:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
11003:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
11004:   {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
11005:   ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
11006:   ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
11007:   ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
11008:   ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
11009:   ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
11010:   ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
11011:   ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
11012:   ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
11013:   ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
11014:   ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
11015:   ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
11016:   {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
11017:   ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
11018:   ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
11019:   ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
11020:   ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
11021:   ('EOAC_NONE','LongWord').SetUInt( $0);
11022:   ('EOAC_DEFAULT','LongWord').SetUInt( $800);
11023:   ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
11024:   ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
11025:   ('EOAC_DYNAMIC_CLOACKING','LongWord').SetUInt( $40);
11026:   ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
11027:   ('RPC_C_AUTHN_WINNT','LongInt'( 10);
11028:   ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
11029:   ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
11030:   ('RPC_C_AUTHNZ_DCE','LongInt'( 2);

```

```

11031:   FindClass ('TOBJECT'), 'EBoldCom
11032:   Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
11033:   Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
11034:   Function BoldStreamToVariant( Stream : TStream) : OleVariant
11035:   Function BoldStringsToVariant( Strings : TStrings) : OleVariant
11036:   Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
11037:   Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
11038:   Function BoldVariantArrayOfArraysOfStringToStrings(V:OleVariant;Strings: TStrings) : Integer
11039:   Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
11040:   Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11041:   Function BoldGetValue( Data : OleVariant; const Name : string) : OleVariant
11042:   Procedure BoldSetValue( Data : OleVariant; const Name : string; Value : OleVariant)
11043:   Function BoldCreateGUID : TGUID
11044:   Function BoldCreateComObject( const ClsId, IID : TGUID; out Obj : variant; out Res : HResult) : Boolean
11045:   Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
Res:HRes):Bool;
11046:   Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11047:   Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11048: end;
11049:
11050: (*-----*)
11051: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11052: begin
11053:   Function ParseISODate( s : string) : TDateTime
11054:   Function ParseISODateTime( s : string) : TDateTime
11055:   Function ParseISOTime( str : string) : TDateTime
11056: end;
11057:
11058: (*-----*)
11059: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11060: begin
11061:   Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
11062:   Function BoldCreateGUIDWithBracketsAsString : string
11063: end;
11064:
11065: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11066: begin
11067:   FindClass ('TOBJECT'), 'TBoldFileHandler
11068:   FindClass ('TOBJECT'), 'TBoldDiskfileHandler
11069:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11070:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
11071:   SIRegister_TBoldFileHandler(CL);
11072:   SIRegister_TBoldDiskFileHandler(CL);
11073:   Procedure BoldCloseAllFilehandlers
11074:   Procedure BoldRemoveUnchangedFilesFromEditor
11075:   Function BoldFileHandlerList : TBoldObjectArray
11076:   Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11077: end;
11078:
11079: procedure SIRegister_BoldWinInet(CL: TPSPascalCompiler);
11080: begin
11081:   PCharArr', 'array of PChar
11082:   Function BoldInternetOpen(Agent:String;
AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr;
11083:   Function BoldInternetOpenUrl(iNet:Pointer;URL:string; Headers:String;Flags,Context:cardinal):Pointer
11084:   Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
NumberOfBytesRead:Card):LongBool;
11085:   Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11086:   Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
Cardinal; Reserved : Cardinal) : LongBool
11087:   Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
Cardinal; Context : Cardinal) : LongBool
11088:   Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
: PCharArr; Flags, Context : Cardinal) : Pointer
11089:   Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11090:   Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11091:   Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11092:   Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11093:   Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents) :BOOL;
11094: end;
11095:
11096: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11097: begin
11098:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11099:   SIRegister_TfrmBoldQueryUser(CL);
11100:   Function QueryUser( const Title, Query : string) : TBoldQueryResult
11101: end;
11102:
11103: (*-----*)
11104: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11105: begin
11106:   //('befIsInDisplayList',' BoldElementFlag0 );
11107:   //('befStronglyDependedOfPrioritized',' BoldElementFlag1 );
11108:   //('befFollowerSelected',' BoldElementFlag2 );
11109:   FindClass ('TOBJECT'), 'TBoldQueue
11110:   FindClass ('TOBJECT'), 'TBoldQueueable
11111:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )

```

```

11112:  SIRегистер_TBoldQueueable(CL);
11113:  SIRегистер_TBoldQueue(CL);
11114:  Function BoldQueueFinalized : Boolean
11115:  Function BoldInstalledQueue : TBoldQueue
11116: end;
11117:
11118: procedure SIRегистер_Barcodes(CL: TPSPascalCompiler);
11119: begin
11120:  const mmPerInch','Extended').setExtended( 25.4);
11121:  TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,
11122:    +'bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11123:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11124:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11125:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11126:  TBarcodeLineType', '( white, black, black_half )
11127:  TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11128:  TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11129:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11130:  TCheckSumMethod', '( csmNone, csmModulo10 )
11131:  SIRегистер_TASBarcode(CL);
11132:  Function CheckSumModulo10( const data : string ) : string
11133:  Function ConvertMmToPixelsX( const Value : Double ) : Integer
11134:  Function ConvertMmToPixelsY( const Value : Double ) : Integer
11135:  Function ConvertInchToPixelsX( const Value : Double ) : Integer
11136:  Function ConvertInchToPixelsY( const Value : Double ) : Integer
11137: end;
11138:
11139: procedure SIRегистер_Geometry(CL: TPSPascalCompiler); //OpenGL
11140: begin
11141:  THomogeneousByteVector', 'array[0..3] of Byte
11142:  THomogeneousWordVector', 'array[0..3] of Word
11143:  THomogeneousIntVector', 'array[0..3] of Integer
11144:  THomogeneousFltVector', 'array[0..3] of single
11145:  THomogeneousDblVector', 'array[0..3] of double
11146:  THomogeneousExtVector', 'array[0..3] of extended
11147:  TAffineByteVector', 'array[0..2] of Byte
11148:  TAffineWordVector', 'array[0..2] of Word
11149:  TAffineIntVector', 'array[0..2] of Integer
11150:  TAffineFltVector', 'array[0..2] of single
11151:  TAffineDblVector', 'array[0..2] of double
11152:  TAffineExtVector', 'array[0..2] of extended
11153:  THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11154:  THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11155:  THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11156:  THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11157:  THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11158:  THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11159:  TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11160:  TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11161:  TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11162:  TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11163:  TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11164:  TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11165:  TMatrix4b', 'THomogeneousByteMatrix
11166:  TMatrix4w', 'THomogeneousWordMatrix
11167:  TMatrix4i', 'THomogeneousIntMatrix
11168:  TMatrix4f', 'THomogeneousFltMatrix
11169:  TMatrix4d', 'THomogeneousDblMatrix
11170:  TMatrix4e', 'THomogeneousExtMatrix
11171:  TMatrix3b', 'TAffineByteMatrix
11172:  TMatrix3w', 'TAffineWordMatrix
11173:  TMatrix3i', 'TAffineIntMatrix
11174:  TMatrix3f', 'TAffineFltMatrix
11175:  TMatrix3d', 'TAffineDblMatrix
11176:  TMatrix3e', 'TAffineExtMatrix
11177:  //PMatrix', '^TMatrix // will not work
11178:  TMatrixGL', 'THomogeneousFltMatrix
11179:  THomogeneousMatrix', 'THomogeneousFltMatrix
11180:  TAffineMatrix', 'TAffineFltMatrix
11181:  Quaternion', 'record Vector : TVector4f; end
11182:  TRectangle', 'record Left : integer; Top : integer; Width : integer;
11183:    +Height : Integer; end
11184:  TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11185:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11186:    +' , ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11187:  'EPSILON','Extended').setExtended( 1E-100 );
11188:  'EPSILON2','Extended').setExtended( 1E-50 );
11189:  Function VectorAddGL( V1, V2 : TVectorGL ) : TVectorGL
11190:  Function VectorAffineAdd( V1, V2 : TAffineVector ) : TAffineVector
11191:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11192:  Function VectorAffineDotProduct( V1, V2 : TAffineVector ) : Single
11193:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single ) : TAffineVector
11194:  Function VectorAffineSubtract( V1, V2 : TAffineVector ) : TAffineVector
11195:  Function VectorAngle( V1, V2 : TAffineVector ) : Single
11196:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single ) : TVectorGL
11197:  Function VectorCrossProduct( V1, V2 : TAffineVector ) : TAffineVector
11198:  Function VectorDotProduct( V1, V2 : TVectorGL ) : Single
11199:  Function VectorLength( V : array of Single ) : Single
11200:  Function VectorLerp( V1, V2 : TVectorGL; t : Single ) : TVectorGL

```

```

11201: Procedure VectorNegate( V : array of Single)
11202: Function VectorNorm( V : array of Single) : Single
11203: Function VectorNormalize( V : array of Single) : Single
11204: Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11205: Function VectorReflect( V, N : TAffineVector) : TAffineVector
11206: Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11207: Procedure VectorScale( V : array of Single; Factor : Single)
11208: Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11209: Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11210: Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11211: Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11212: Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11213: Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11214: Procedure MatrixAdjoint( var M : TMatrixGL)
11215: Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11216: Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11217: Function MatrixDeterminant( M : TMatrixGL) : Single
11218: Procedure MatrixInvert( var M : TMatrixGL)
11219: Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11220: Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11221: Procedure MatrixTranspose( var M : TMatrixGL)
11222: Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11223: Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11224: Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11225: Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11226: Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11227: Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11228: Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11229: Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11230: //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11231: Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11232: Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11233: Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11234: Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11235: Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11236: Function MakeAffineVector( V : array of Single) : TAffineVector
11237: Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11238: Function MakeVector( V : array of Single) : TVectorGL
11239: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11240: Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11241: Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11242: Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11243: Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11244: Function ArcCosGL( X : Extended) : Extended
11245: Function ArcSingGL( X : Extended) : Extended
11246: Function ArcTan2GL( Y, X : Extended) : Extended
11247: Function CoTanGL( X : Extended) : Extended
11248: Function DegToRadGL( Degrees : Extended) : Extended
11249: Function RadToDegGL( Radians : Extended) : Extended
11250: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11251: Function TanGL( X : Extended) : Extended
11252: Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11253: Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11254: Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11255: Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11256: Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11257: Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11258: end;
11259:
11260:
11261: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11262: begin
11263:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11264:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11265:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11266:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11267:   Function RegReadBoolDef(const RootKey:HKEY; const Key, Name: string; Def: Boolean) : Boolean
11268:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11269:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11270:   Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11271:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11272:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11273:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11274:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11275:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11276:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11277:   Procedure RegWriteWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11278:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11279:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11280:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11281:   Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11282:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11283:             +'RunOnce, ekServiceRun, ekServiceRunOnce )'
11284:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError'
11285:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11286:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11287:   Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
11288: Items:TStrings):Bool;
11288:   Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
11288: SaveTo:TStrings):Bool;
```

```

11289: Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11290: end;
11291:
11292: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11293: begin
11294:   CLSID_StdComponentCategoriesMgr', 'TGUID '{0002E005-0000-0000-C000-000000000046}
11295:   CATID_SafeForInitializing', 'TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11296:   CATID_SafeForScripting', 'TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11297:   icMAX_CATEGORY_DESC_LEN', 'LongInt' ( 128);
11298:   FindClass ('TOBJECT'), 'EInvalidParam
11299:   Function IsDCOMInstalled : Boolean
11300:   Function IsDCOMEabled : Boolean
11301:   Function GetDCOMVersion : string
11302:   Function GetMDACVersion : string
11303:   Function GetMDACVersion2 : string
11304:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11305:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11306:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11307:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11308:   Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11309:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11310:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11311:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11312:   Function ResetIStreamToStart( Stream : IStream) : Boolean
11313:   Function SizeOfIStreamContents( Stream : IStream) : Largeint
11314:   Function StreamToVariantArray( Stream : TStream) : OleVariant;
11315:   Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11316:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11317:   Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11318: end;
11319:
11320:
11321: procedure SIRegister_JclUnitConv_mx2(CL: TPSPascalCompiler);
11322: begin
11323:   Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11324:   FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11325:   KelvinFreezingPoint','Extended').setExtended( 273.15);
11326:   CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11327:   FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11328:   KelvinAbsoluteZero','Extended').setExtended( 0.0);
11329:   DegPerCycle','Extended').setExtended( 360.0);
11330:   DegPerGrad','Extended').setExtended( 0.9);
11331:   DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11332:   GradPerCycle','Extended').setExtended( 400.0);
11333:   GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11334:   GradPerRad','Extended').setExtended( 63.66197236758134307553505349006);
11335:   RadPerCycle','Extended').setExtended( 6.283185307179586476925286766559);
11336:   RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11337:   RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11338:   CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777777778);
11339:   CyclePerGrad','Extended').setExtended( 0.0025);
11340:   CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11341:   ArcMinutesPerDeg','Extended').setExtended( 60.0);
11342:   ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11343:   Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11344:   Function MakePercentage( const Step, Max : Longint) : Longint
11345:   Function CelsiusToKelvin( const T : double) : double
11346:   Function CelsiusToFahrenheit( const T : double) : double
11347:   Function KelvinToCelsius( const T : double) : double
11348:   Function KelvinToFahrenheit( const T : double) : double
11349:   Function FahrenheitToCelsius( const T : double) : double
11350:   Function FahrenheitToKelvin( const T : double) : double
11351:   Function CycleToDeg( const Cycles : double) : double
11352:   Function CycleToGrad( const Cycles : double) : double
11353:   Function CycleToRad( const Cycles : double) : double
11354:   Function DegToCycle( const Degrees : double) : double
11355:   Function DegToGrad( const Degrees : double) : double
11356:   Function DegToRad( const Degrees : double) : double
11357:   Function GradToCycle( const Grads : double) : double
11358:   Function GradToDeg( const Grads : double) : double
11359:   Function GradToRad( const Grads : double) : double
11360:   Function RadToCycle( const Radians : double) : double
11361:   Function RadToDeg( const Radians : double) : double
11362:   Function RadToGrad( const Radians : double) : double
11363:   Function DmsToDeg( const D, M : Integer; const S : double) : double
11364:   Function DmsToRad( const D, M : Integer; const S : double) : double
11365:   Procedure DgToDms( const Degrees : double; out D, M : Integer; out S : double)
11366:   Function DgToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11367:   Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11368:   Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11369:   Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11370:   Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11371:   Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11372:   Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11373:   Function CmToInch( const Cm : double) : double
11374:   Function InchToCm( const Inch : double) : double

```

```

11375: Function FeetToMetre( const Feet : double) : double
11376: Function MetreToFeet( const Metre : double) : double
11377: Function YardToMetre( const Yard : double) : double
11378: Function MetreToYard( const Metre : double) : double
11379: Function NmToKm( const Nm : double) : double
11380: Function KmToNm( const Km : double) : double
11381: Function KmToSm( const Km : double) : double
11382: Function SmToKm( const Sm : double) : double
11383: Function LitreToGalUs( const Litre : double) : double
11384: Function GalUsToLitre( const GalUs : double) : double
11385: Function GalUsToGalCan( const GalUs : double) : double
11386: Function GalCanToGalUs( const GalCan : double) : double
11387: Function GalUsToGalUk( const GalUs : double) : double
11388: Function GalUkToGalUs( const GalUk : double) : double
11389: Function LitreToGalCan( const Litre : double) : double
11390: Function GalCanToLitre( const GalCan : double) : double
11391: Function LitreToGalUk( const Litre : double) : double
11392: Function GalUkToLitre( const GalUk : double) : double
11393: Function KgToLb( const Kg : double) : double
11394: Function LbToKg( const Lb : double) : double
11395: Function KgToOz( const Kg : double) : double
11396: Function OzToKg( const Oz : double) : double
11397: Function CwtUsToKg( const Cwt : double) : double
11398: Function CwtUkToKg( const Cwt : double) : double
11399: Function KaratToKg( const Karat : double) : double
11400: Function KgToCwtUs( const Kg : double) : double
11401: Function KgToCwtUk( const Kg : double) : double
11402: Function KgToKarat( const Kg : double) : double
11403: Function KgToSton( const Kg : double) : double
11404: Function KgToLton( const Kg : double) : double
11405: Function StonToKg( const STon : double) : double
11406: Function LtonToKg( const Lton : double) : double
11407: Function QrUsToKg( const Qr : double) : double
11408: Function QrUkToKg( const Qr : double) : double
11409: Function KgToQrUs( const Kg : double) : double
11410: Function KgToQrUk( const Kg : double) : double
11411: Function PascalToBar( const Pa : double) : double
11412: Function PascalToAt( const Pa : double) : double
11413: Function PascalToTorr( const Pa : double) : double
11414: Function BarToPascal( const Bar : double) : double
11415: Function AtToPascal( const At : double) : double
11416: Function TorrToPascal( const Torr : double) : double
11417: Function KnotToMs( const Knot : double) : double
11418: Function HpElectricToWatt( const HpE : double) : double
11419: Function HpMetricToWatt( const HpM : double) : double
11420: Function MsToKnot( const ms : double) : double
11421: Function WattToHpElectric( const W : double) : double
11422: Function WattToHpMetric( const W : double) : double
11423: function getBigPI: string; //PI of 1000 numbers
11424:
11425: procedure SIRegister_devutils(CL: TPSPascalCompiler);
11426: begin
11427:   Function CDEexecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11428:   Procedure CDCopyFile( const FileName, DestName : string)
11429:   Procedure CDMoveFile( const FileName, DestName : string)
11430:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11431:   Procedure CDDeleteFiles( Sender : TObject; s : string)
11432:   Function CDGetTempDir : string
11433:   Function CDGetFileSize( FileName : string) : longint
11434:   Function GetFileTime( FileName : string) : longint
11435:   Function GetShortName( FileName : string) : string
11436:   Function GetFullName( FileName : string) : string
11437:   Function WinReboot : boolean
11438:   Function Windir : String
11439:   Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11440:   Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11441:   Function devExecutor : TdevExecutor
11442: end;
11443:
11444: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11445: begin
11446:   Procedure CheckAssociations // AssociationsCount', 'LongInt'( 7);
11447:   Procedure Associate( Index : integer)
11448:   Procedure UnAssociate( Index : integer)
11449:   Function IsAssociated( Index : integer) : boolean
11450:   Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11451:   Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp, IcoNum: string)
11452:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11453:   procedure RefreshIcons;
11454:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11455:   function MergColor(Colors: Array of TColor): TColor;
11456:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11457:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11458:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11459:   function GetInverseColor(AColor: TColor): TColor;
11460:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11461:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: integer; ShadowColor: TColor);
11462:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11463:   Procedure GetSystemMenuFont(Font: TFont);

```

```

11464: end;
11465:
11466: //*****unit uPSI_JvHLPParser;*****
11467: function IsStringConstant(const St: string): Boolean;
11468: function IsIntConstant(const St: string): Boolean;
11469: function IsRealConstant(const St: string): Boolean;
11470: function IsIdentifier(const ID: string): Boolean;
11471: function GetStringValue(const St: string): string;
11472: procedure ParseString(const S: string; Ss: TStrings);
11473: function IsStringConstantW(const St: WideString): Boolean;
11474: function IsIntConstantW(const St: WideString): Boolean;
11475: function IsRealConstantW(const St: WideString): Boolean;
11476: function IsIdentifierW(const ID: WideString): Boolean;
11477: function GetStringValueW(const St: WideString): WideString;
11478: procedure ParseStringW(const S: WideString; Ss: TStrings);
11479:
11480:
11481: //*****unit uPSI_JclMapi;*****
11482:
11483: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment : TFileName; ShowDialog : Boolean; AParentWND : HWnd ) : Boolean;
11484: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment : TFileName; ShowDialog : Boolean; AParentWND : HWnd ) : Boolean;
11485: Function JclSimpleBringUpSendMailDialog(const ASubject, ABody:string;const AAttach:TFileName;AParentWND:HWND):Bool;
11486: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean ) : DWORD;
11487: Function MapiErrorMessage( const ErrorCode : DWORD ) : string;
11488:
11489: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11490: begin
11491:   //'pdes_key_schedule', '^des_key_schedule // will not work
11492:   Function BuildType1Message( ADomain, AHost : String ) : String;
11493:   Function BuildType3Message( ADomain, AHost, AUsername:WideString; APassword, ANonce:String ):String;
11494:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIIdAuthenticationClass );
11495:   Function FindAuthClass( AuthName : String ) : TIIdAuthenticationClass;
11496:   GBase64CodeTable', 'string'ABCDEFGHJKLMNOPQRSTUVWXYZabcdefgijklmnopqrstuvwxyz0123456789+/
11497:   GXCECodeTable', 'string'+0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefgijklmnopqrstuvwxyz
11498:   GUUECodeTable', 'string`'#$%&'()'*,.-./0123456789:;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_
11499: end;
11500:
11501: procedure SIRegister_WDOSocketUtils(CL: TPSPascalCompiler);
11502: begin
11503:   ('IpAny', 'LongWord').SetUInt( $00000000 );
11504:   IpLoopBack', 'LongWord').SetUInt( $7F000001 );
11505:   IpBroadcast', 'LongWord').SetUInt( $FFFFFFFF );
11506:   IpNone', 'LongWord').SetUInt( $0000 );
11507:   PortAny', 'LongWord( $0000 );
11508:   SocketMaxConnections', 'LongInt'( 5 );
11509:   TIPAddr', 'LongWord
11510:   TIPRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11511:   Function HostToNetLong( HostLong : LongWord ) : LongWord;
11512:   Function HostToNetShort( HostShort : Word ) : Word;
11513:   Function NetToHostLong( NetLong : LongWord ) : LongWord;
11514:   Function NetToHostShort( NetShort : Word ) : Word;
11515:   Function StrToIP( IP : string ) : TIPAddr;
11516:   Function IPToStr( IP : TIPAddr ) : string;
11517: end;
11518:
11519: (*-----*)
11520: procedure SIRegister_ALSMTPCClient(CL: TPSPascalCompiler);
11521: begin
11522:   TAISmtPCClientAuthType', '( AlSmtpClientAuthNone, alSmtpClientAuth'
11523:   +'hPlain, AlSmtpClientAuthLogin, AlSmtpClientAuthCramMD5, AlSmtpClientAuthCr'
11524:   +'amShal, AlSmtpClientAuthAutoSelect );
11525:   TAISmtPCClientAuthTypeSet', 'set of TAISmtPCClientAuthType
11526:   SIRegister_TAISmtPCClient(CL);
11527: end;
11528:
11529: procedure SIRegister_WDOSPlcUtils(CL: TPSPascalCompiler);
11530: begin
11531:   'TBitNo', 'Integer
11532:   TStByteNo', 'Integer
11533:   TStationNo', 'Integer
11534:   TInOutNo', 'Integer
11535:   TIo', '( EE, AA, NE, NA )
11536:   TBitSet', 'set of TBitNo
11537:   TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11538:   TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11539:   TBitAddr', 'LongInt
11540:   TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11541:   TByteAddr', 'SmallInt
11542:   TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11543:   Function BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr;
11544:   Function BusBitAddr(aIo: TIO;aInOutNo:TInOutNo;aStat:TStatNo;aStatByteNo:TStatByteNo;aBitNo:TBitNo):TBitAddr;
11545:   Procedure BitAddrToValues(aBitAddr:TBitAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var aBitNo:TBitNo);
11546:   Function BitAddrToStr( Value : TBitAddr ) : string;
11547:   Function StrToBitAddr( const Value : string ) : TBitAddr;
11548:   Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte ) : TByteAddr;

```

```

11549: Function BusByteAddr(aIo:TIO;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11550: Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11551: Function ByteAddrToStr( Value : TByteAddr ) : string
11552: Function StrToByteAddr( const Value : string ) : TByteAddr
11553: Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11554: Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11555: Function InOutStateToStr( State : TInOutState ) : string
11556: Function MasterErrorToStr( ErrorCode : TErrorCode ) : string
11557: Function SlaveErrorToStr( ErrorCode : TErrorCode ) : string
11558: end;
11559:
11560: procedure SIRegister_WDOSTimers(CL: TPSPascalCompiler);
11561: begin
11562: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11563:     +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11564: DpmiPmVector', 'Int64
11565: 'DInterval','LongInt'( 1000 );
11566: //{'Enabled','Boolean'}BoolToStr( True );
11567: 'DIntFreq','string' if64
11568: //{'DMessages','Boolean' if64};
11569: SIRegister_TwdxCustomTimer(CL);
11570: SIRegister_TwdxTimer(CL);
11571: SIRegister_TwdxRtcTimer(CL);
11572: SIRegister_TCustomIntTimer(CL);
11573: SIRegister_TIntTimer(CL);
11574: SIRegister_TRtcIntTimer(CL);
11575: Function RealNow : TDateTime
11576: Function MsToDateTime( MilliSecond : LongInt ) : TDateTime
11577: Function DateTimeToMs( Time : TDateTime ) : LongInt
11578: end;
11579:
11580: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11581: begin
11582: TIdSyslogPRI', 'Integer
11583: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11584:     +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11585:     +'kDaemonOne, sfSecurityTwo, sfTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11586:     +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11587:     +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11588: TIdSyslogSeverity', '(slEmergency, slAlert, slCritical, slError, slWarning, slNotice, slInformational, slDebug)
11589: SIRegister_TIdSysLogMsgPart(CL);
11590: SIRegister_TIdSysLogMessage(CL);
11591: Function FacilityToString( AFac : TIdSyslogFacility ) : string
11592: Function SeverityToString( ASec : TIdSyslogSeverity ) : string
11593: Function NoToSeverity( ASev : Word ) : TIdSyslogSeverity
11594: Function logSeverityToNo( ASev : TIdSyslogSeverity ) : Word
11595: Function NoToFacility( AFac : Word ) : TIdSyslogFacility
11596: Function logFacilityToNo( AFac : TIdSyslogFacility ) : Word
11597: end;
11598:
11599: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11600: begin
11601: 'UWhitespace','String '(?:\s*)
11602: Function StripSpaces( const AText : string ) : string
11603: Function CharCount( const AText : string; Ch : Char ) : Integer
11604: Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11605: Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11606: end;
11607:
11608:
11609: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11610: begin
11611: ExtPascalVersion', 'String '0.9.8
11612: AddTypes('TBrower', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11613:     +'Opera, brKonqueror, brMobileSafari )
11614: AddTypes('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11615: AddTypes('TExtProcedure', 'Procedure
11616: Function DetermineBrowser( const UserAgentStr : string ) : TBrower
11617: Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11618: Function ExtExplode( Delim : char; const S : string; Separator : char ) : TStringList
11619: Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11620: Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11621: Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11622: Function StrToJS( const S : string; UseBR : boolean ) : string
11623: Function CaseOf( const S : string; const Cases : array of string ) : integer
11624: Function RCaseOf( const S : string; const Cases : array of string ) : integer
11625: Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer ) : string
11626: Function SetPaddings(Top:integer;Right:int;Bottom:int;Left:int;CSSUnit:TCSSUnit;Header:bool):string;
11627: Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11628: Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11629: Function IsUpperCase( S : string ) : boolean
11630: Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11631: Function BeautifyCSS( const AStyle : string ) : string
11632: Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11633: Function JSDateToDate( JSDate : string ) : TDateTime
11634: end;
11635:
11636: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11637: begin

```

```

11638: TSHDeleteOption', '(& doSilent, doAllowUndo, doFilesOnly )
11639: TSHDeleteOptions', 'set of TSHDeleteOption
11640: TSHRenameOption', '(& roSilent, roRenameOnCollision )
11641: TSHRenameOptions', 'set of TSHRenameOption
11642: Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11643: Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11644: Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11645: TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11646: TEnumFolderFlags', 'set of TEnumFolderFlag
11647: TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR
11648: +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11649: +'IEnumIdList; Folder : IShellFolder; end
11650: Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11651: Function SHEnumSpecialFolderFirst(SpecFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11652: Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11653: Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11654: Function GetSpecialFolderLocation( const Folder : Integer) : string
11655: Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11656: Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11657: Function DisplayContextMenu(const Handle:HWND; const FileName: string; Pos:TPoint): Boolean
11658: Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11659: Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11660: Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11661: Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11662: Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11663: Function SHFreeMem( var P : Pointer) : Boolean
11664: Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11665: Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11666: Function PathToPidlBind( const PathName : string; out Folder : IShellFolder) : PItemIdList
11667: Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11668: Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11669: Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11670: Function PidlFree( var IdList : PItemIdList) : Boolean
11671: Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11672: Function PidlGetLength( const Pidl : PItemIdList) : Integer
11673: Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11674: Function PidlToPath( Idlist : PItemIdList) : string
11675: Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11676: Function StrRetToString( Idlist : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11677: PShellLink', '^TShellLink // will not work
11678: TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11679: +'ingDirectory : string; IdList : PItemIdList; Target : string; Description '
11680: +' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11681: Procedure ShellLinkFree( var Link : TShellLink)
11682: Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11683: Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11684: Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11685: Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11686: Function SHD11GetVersion( const FileName : string; var Version : TD11VersionInfo) : Boolean
11687: Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11688: Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11689: Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11690: Function OverlayIconShared( var Large, Small : HICON) : Boolean
11691: Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11692: Function ShellExecEx(const FileName:string;const Paramtrs:string;const Verb:string; CmdShow:Int):Bool;
11693: Function ShellExec(Wnd: Integer;const Operati,FileName,Paramtrs,Directy:string;ShowCommand:Int):Bool;
11694: Function ShellExecAndWait(const FileName:string;const Params:string;const Verb:string;CmdShow:Int):Bool;
11695: Function ShellOpenAs( const FileName : string) : Boolean
11696: Function ShellRasDial( const EntryName : string) : Boolean
11697: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11698: Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11699: TJclFileExeType', '(& etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11700: Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11701: Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11702: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11703: Function OemKeyScan( wOemChar : Word) : DWORD
11704: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11705: end;
11706:
11707: procedure SIRegister_cXMLFunctions(CL: TPPascalCompiler);
11708: begin
11709: xmlVersion', 'String '1.0 FindClass('TOBJECT'),'Exml
11710: //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11711: Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11712: Function xmlValidChar2( const Ch : WideChar) : Boolean;
11713: Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11714: Function xmlIsLetter( const Ch : WideChar) : Boolean
11715: Function xmlIsDigit( const Ch : WideChar) : Boolean
11716: Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11717: Function xmlIsNameChar( const Ch : WideChar) : Boolean
11718: Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11719: Function xmlValidName( const Text : UnicodeString) : Boolean
11720: //xmlSpace', 'Char #$20 or #$9 or #$D or #$A);
11721: //Function xmlSkipSpace( var P : PWideChar) : Boolean
11722: //Function xmlSkipEq( var P : PWideChar) : Boolean
11723: //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11724: //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
: TUnicodeCodecClass
11725: Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar

```

```

11726: Function xmlTag( const Tag : UnicodeString) : UnicodeString
11727: Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11728: Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11729: Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11730: Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11731: Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11732: Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11733: Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11734: Function xmlComment( const Comment : UnicodeString) : UnicodeString
11735: Procedure SelfTestcXMLFunctions
11736: end;
11737:
11738: (*-----*)
11739: procedure SIRegister_DepWalkUtils(CL: TPSPPascalCompiler);
11740: begin
11741:   Function AWaitCursor : IUnknown
11742:   Function ChangeCursor( NewCursor : TCursor) : IUnknown
11743:   Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11744:   Function YesNo( const ACaption, AMsg : string) : boolean
11745:   Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11746:   Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11747:   Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11748:   Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11749:   Procedure GetSystemPaths( Strings : TStrings)
11750:   Procedure MakeEditNumeric( EditHandle : integer)
11751: end;
11752:
11753: procedure SIRegister_yuvconverts(CL: TPSPPascalCompiler);
11754: begin
11755:   AddTypes('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11756:   'BI_YUY2','LongWord($32595559);
11757:   'BI_UYVY','LongWord').SetUInt($59565955);
11758:   'BI_BTYUV','LongWord').SetUInt($50313459);
11759:   'BI_YVU9','LongWord').SetUInt($39555659);
11760:   'BI_YUV12','LongWord($30323449);
11761:   'BI_Y8','LongWord').SetUInt($20203859);
11762:   'BI_Y211','LongWord').SetUInt($31313259);
11763:   Function BICompressionToVideoCodec( Value: DWord) : TVideoCodec
11764:   Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11765: end;
11766:
11767: (*-----*)
11768: procedure SIRegister_AviCap(CL: TPSPPascalCompiler);
11769: begin
11770:   'WM_USER','LongWord').SetUInt($0400);
11771:   'WM_CAP_START','LongWord').SetUInt($0400);
11772:   'WM_CAP_END','longword').SetUInt($0400+85);
11773: //WM_CAP_START+ 85
11774: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11775:   Function capSetCallbackOnErrorHandler( hwnd : THandle; fpProc : LongInt) : LongInt
11776:   Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11777:   Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11778:   Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11779:   Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11780:   Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11781:   Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11782:   Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11783:   Function capGetUserData( hwnd : THandle) : LongInt
11784:   Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11785:   Function capDriverDisconnect( hwnd : THandle) : LongInt
11786:   Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11787:   Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11788:   Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11789:   Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11790:   Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11791:   Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11792:   Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11793:   Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11794:   Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11795:   Function capEditCopy( hwnd : THandle) : LongInt
11796:   Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11797:   Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11798:   Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11799:   Function capDlgVideoFormat( hwnd : THandle) : LongInt
11800:   Function capDlgVideoSource( hwnd : THandle) : LongInt
11801:   Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11802:   Function capDlgVideoCompression( hwnd : THandle) : LongInt
11803:   Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11804:   Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11805:   Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11806:   Function capPreview( hwnd : THandle; f : Word) : LongInt
11807:   Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11808:   Function capOverlay( hwnd : THandle; f : Word) : LongInt
11809:   Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11810:   Function capGetStatus( hwnd : THandle; s : LongInt; wsize : Word) : LongInt
11811:   Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11812:   Function capGrabFrame( hwnd : THandle) : LongInt
11813:   Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11814:   Function capCaptureSequence( hwnd : THandle) : LongInt

```

```

11815: Function capCaptureSequenceNoFile( hwnd : THandle ) : LongInt
11816: Function capCaptureStop( hwnd : THandle ) : LongInt
11817: Function capCaptureAbort( hwnd : THandle ) : LongInt
11818: Function capCaptureSingleFrameOpen( hwnd : THandle ) : LongInt
11819: Function capCaptureSingleFrameClose( hwnd : THandle ) : LongInt
11820: Function capCaptureSingleFrame( hwnd : THandle ) : LongInt
11821: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11822: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11823: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt ) : LongInt
11824: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word ) : LongInt
11825: Function capPaletteOpen( hwnd : THandle; szName : LongInt ) : LongInt
11826: Function capPaletteSave( hwnd : THandle; szName : LongInt ) : LongInt
11827: Function capPalettePaste( hwnd : THandle ) : LongInt
11828: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt ) : LongInt
11829: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt ) : LongInt
11830: //PCAPDriverCaps', '^TCapDriverCaps' // will not work
11831: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11832: +' ; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11833: +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11834: +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11835: //PCapStatus', '^TCapStatus' // will not work
11836: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11837: +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT'
11838: +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11839: +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11840: +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11841: +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD; '
11842: +'wNumAudioAllocated : WORD; end
11843: //PCaptureParms', '^TCaptureParms' // will not work
11844: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11845: +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11846: +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11847: +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11848: +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11849: +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11850: +'wMCICStartTime : DWORD; dwMCICStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11851: +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11852: +'he : BOOL; AVStreamMaster : WORD; end
11853: // PCapInfoChunk', '^TCapInfoChunk' // will not work
11854: //TCapInfoChunk', 'record fccInfoId : FOURCC; lpData : LongInt; cbData : LongInt; end
11855: 'CONTROLCALLBACK_PREROLL','LongInt'( 1 );
11856: 'CONTROLCALLBACK_CAPTURING','LongInt'( 2 );
11857: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nID : Integer) : THandle
11858: Function
capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool,
11859: 'IDS_CAP_BEGIN','LongInt'( 300 );
11860: 'IDS_CAP_END','LongInt'( 301 );
11861: 'IDS_CAP_INFO','LongInt'( 401 );
11862: 'IDS_CAP_OUTOFGMEM','LongInt'( 402 );
11863: 'IDS_CAP_FILEEXISTS','LongInt'( 403 );
11864: 'IDS_CAP_ERRORPALOPEN','LongInt'( 404 );
11865: 'IDS_CAP_ERRORPALSAVE','LongInt'( 405 );
11866: 'IDS_CAP_ERRORDIBSAVE','LongInt'( 406 );
11867: 'IDS_CAP_DEFAVIEXT','LongInt'( 407 );
11868: 'IDS_CAP_DEFFALEXT','LongInt'( 408 );
11869: 'IDS_CAP_CANTOPEN','LongInt'( 409 );
11870: 'IDS_CAP_SEQ_MSGSTART','LongInt'( 410 );
11871: 'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411 );
11872: 'IDS_CAP_VIDEEDITERR','LongInt'( 412 );
11873: 'IDS_CAP_READONLYFILE','LongInt'( 413 );
11874: 'IDS_CAP_WRITEERROR','LongInt'( 414 );
11875: 'IDS_CAP_NODISKSPACE','LongInt'( 415 );
11876: 'IDS_CAP_SETFILESIZE','LongInt'( 416 );
11877: 'IDS_CAP_SAVEASPERCENT','LongInt'( 417 );
11878: 'IDS_CAP_DRIVER_ERROR','LongInt'( 418 );
11879: 'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419 );
11880: 'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420 );
11881: 'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421 );
11882: 'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422 );
11883: 'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423 );
11884: 'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424 );
11885: 'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425 );
11886: 'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426 );
11887: 'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427 );
11888: 'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428 );
11889: 'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429 );
11890: 'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430 );
11891: 'IDS_CAP_RECORDING_ERROR','LongInt'( 431 );
11892: 'IDS_CAP_RECORDING_ERROR2','LongInt'( 432 );
11893: 'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433 );
11894: 'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434 );
11895: 'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435 );
11896: 'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436 );
11897: 'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437 );
11898: 'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438 );
11899: 'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439 );
11900: 'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440 );
11901: 'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441 );

```

```

11902:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11903:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11904:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11905:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11906:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11907:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11908:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11909:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11910:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11911:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11912:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11913:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11914:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11915:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11916:  'AVICAP32','String 'AVICAP32.dll
11917: end;
11918:
11919: procedure SIRegister_ALFcnsMisc(CL: TPSPPascalCompiler);
11920: begin
11921:   Function AlBoolToInt( Value : Boolean ) : Integer
11922:   Function ALMediumPos( LTotal, LBorder, LObject : integer ) : Integer
11923:   Function AlIsValidEmail( const Value : AnsiString ) : boolean
11924:   Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TdateTime ) : TdateTime
11925:   Function ALInc( var x : integer; Count : integer ) : Integer
11926:   function ALCopyStr(const aSourceString:AnsiString;aStart,aLength: Integer): AnsiString
11927:   function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word= fmShareDenyWrite):AnsiString;
11928:   procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11929:   Function ALIsInteger(const S: AnsiString): Boolean;
11930:   function ALIsDecimal(const S: AnsiString): boolean;
11931:   Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11932:   function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11933:   function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''): AnsiString;
11934:   function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11935:   function ALUTF8removeBOM(const S: AnsiString): AnsiString;
11936:   Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11937:   Function ALRandomStr(const aLength: Longint): AnsiString;
11938:   Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11939:   Function ALRandomStrU(const aLength: Longint): String;
11940: end;
11941:
11942: procedure SIRegister_ALJSONDoc(CL: TPSPPascalCompiler);
11943: begin
11944:   Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const aTrueStr: AnsiString; const aFalseStr : AnsiString)
11945: end;
11946:
11947: procedure SIRegister_ALWindows(CL: TPSPPascalCompiler);
11948: begin
11949:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11950:   +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11951:   +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11952:   +'; ullAvailExtendedVirtual : Int64; end
11953:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11954:   Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX ) : BOOL
11955:   Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11956:   'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11957:   'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2 );
11958:   'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1 );
11959:   'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8 );
11960:   'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4 );
11961: end;
11962:
11963: procedure SIRegister_IPCThrd(CL: TPSPPascalCompiler);
11964: begin
11965:   SIRegister_THandledObject(CL);
11966:   SIRegister_TEvent(CL);
11967:   SIRegister_TMutex(CL);
11968:   SIRegister_TSharedMem(CL);
11969:   'TRACE_BUF_SIZE','LongInt'( 200 * 1024 );
11970:   'TRACE_BUFFER','String 'TRACE_BUFFER
11971:   'TRACE_MUTEX','String 'TRACE_MUTEX
11972:   //PTTraceEntry', 'TTraceEntry // will not work
11973:   SIRegister_TIPCTracer(CL);
11974:   'MAX_CLIENTS','LongInt'( 6 );
11975:   'IPCTIMEOUT','LongInt'( 2000 );
11976:   'IPCBUFFER_NAME','String 'BUFFER_NAME
11977:   'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11978:   'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11979:   'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11980:   'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11981:   'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11982:   'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11983:   FindClass('TOBJECT'),'EMonitorActive
11984:   FindClass('TOBJECT'),'TIPCThread
11985:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11986:   +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11987:   +'ach, evClientSwitch, evClientSignal, evClientExit )
11988:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11989:   TClientFlags', 'set of TClientFlag

```

```

1190:  //PEventData', '^TEventData // will not work
1191:  TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
1192:    +'lag; Flags : TClientFlags; end
1193:  TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
1194:  TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
1195:  TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
1196:  //PIPCEventInfo', '^TIPCEventInfo // will not work
1197:  TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
1198:  SIRegister_TIPCEvent(CL);
1199:  //PClientDirRecords', '^TClientDirRecords // will not work
1200:  SIRegister_TClientDirectory(CL);
12001:  TIPCState', '( stInactive, stDisconnected, stConnected )
12002:  SIRegister_TIPCThread(CL);
12003:  SIRegister_TIPCMonitor(CL);
12004:  SIRegister_TIPCCClient(CL);
12005:  Function IsMonitorRunning( var Hndl : THandle ) : Boolean
12006:  end;
12007:
12008:  (*-----*)
12009:  procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
12010: begin
12011:   SIRegister_TALGSMComm(CL);
12012:   Function ALGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString ) : AnsiString
12013:   Procedure ALGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
AMessage:AnsiString);
12014:   Function ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString ) : AnsiString
12015:   Function ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
UseGreekAlphabet:Bool):Widestring;
12016:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
12017:  end;
12018:
12019:  procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
12020: begin
12021:   TALHTTPPropertyChangeEvent', 'Procedure(sender:Tobject;const PropertyIndex:Integer;
12022:   TALHTTPProtocolVersion', '( HTTPPpv_1_0, HTTPPpv_1_1 )
12023:   TALHTTPMethod', '( HTTPPmt_Get,HTTPPmt_Post,HTTPPmt_Head,HTTPPmt_Trace,HTTPPmt_Put,HTTPPmt_Delete );
12024:   TIInternetScheme', 'integer
12025:   TALIPv6Binary', 'array[1..16] of Char;
12026: // TALIPv6Binary = array[1..16] of ansiChar;
12027: // TIInternetScheme = Integer;
12028:   SIRegister_TALHTTPRequestHeader(CL);
12029:   SIRegister_TALHTTPCookie(CL);
12030:   SIRegister_TALHTTPCookieCollection(CL);
12031:   SIRegister_TALHTTPResponseHeader(CL);
12032:   Function ALHTTPDecode( const AStr : AnsiString ) : AnsiString
12033:   Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings )
12034: // Procedure ALEExtractHTTPFields(Separators,WhiteSpace, Quotes:TSysCharSet,
Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
12035: // Procedure ALEExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
12036: // Procedure ALEExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
12037:   Function AlRemoveShemeFromUrl( aUrl : AnsiString ) : ansiString
12038:   Function AlExtractShemeFromUrl( aUrl : AnsiString ) : TIInternetScheme
12039:   Function AlExtractHostNameFromUrl( aUrl : AnsiString ) : AnsiString
12040:   Function AlExtractDomainNameFromUrl( aUrl : AnsiString ) : AnsiString
12041:   Function AlExtractUrlPathFromUrl( aUrl : AnsiString ) : AnsiString
12042:   Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
ExtraInfo : AnsiString; var PortNumber : integer ) : Boolean;
12043:   Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password, UrlPath,
Anchor : AnsiString; Query : TALStrings; var PortNumber : integer ) : Boolean;
12044:   Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Bool;
12045:   Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString ) : AnsiString;
12046:   Function AlRemoveAnchorFromUrl1( aUrl : AnsiString ) : AnsiString;
12047:   Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString ) : AnsiString;
12048:   Function AlCombineUrl1(RelativeUrl,BaseUrl,Anchor:AnsiString;Query:TALStrings): AnsiString;
12049:   Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
12050:   Function ALDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
12051:   Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime ) : Boolean
12052:   Function ALRfc822StrToGMTDateTime( const s : AnsiString ) : TDateTime
12053:   Function ALTryIPV4ToStrToNumeric( aIPV4Str : ansiString; var aIPV4Num : Cardinal ) : Boolean
12054:   Function ALIPV4ToStrToNumeric( aIPV4 : ansiString ) : Cardinal
12055:   Function ALNumericToIPV4Str( aIPV4 : Cardinal ) : ansiString
12056:   Function ALZeroIpV6 : TALIPv6Binary
12057:   Function ALTryIPV6StrToBinary( aIPV6Str : ansiString; var aIPV6Bin : TALIPv6Binary ) : Boolean
12058:   Function ALIPV6StrToBinary( aIPV6 : ansiString ) : TALIPv6Binary
12059:   Function ALBinaryToIPV6Str( aIPV6 : TALIPv6Binary ) : ansiString
12060:   Function ALBinaryStrToIPV6Binary( aIPV6BinaryStr : ansiString ) : TALIPv6Binary
12061: end;
12062:
12063:  procedure SIRegister_ALFcHTML(CL: TPSPascalCompiler);
12064: begin
12065:  Procedure ALUTF8ExtractHTMLText (HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
DecodeHTMLText:Bool;
12066:  Function ALUTF8ExtractHTMLText1 (HtmlContent:AnsiString;const DecodeHTMLText:Boolean) : AnsiString;
12067:  Function ALXMLCDataElementEncode( Src : AnsiString ) : AnsiString
12068:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12069:  Function ALUTF8XMLElementDecode( const Src : AnsiString ) : AnsiString
12070:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
useNumRef:bool):AnsiString;

```

```

12071: Function ALUTF8HTMLDecode( const Src : AnsiString ) : AnsiString
12072: Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean ) : AnsiString
12073: Function ALUTF8JavascriptDecode( const Src : AnsiString ) : AnsiString
12074: Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12075: Procedure ALCompactHtmlTagParams( TagParams : TALStrings )
12076: end;
12077:
12078: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12079: begin
12080:   SIRegister_TALEMailHeader(CL);
12081:   SIRegister_TALNewsArticleHeader(CL);
12082:   Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
decodeRealName:Bool):AnsiString;
12083:   Function AlExtractEmailAddress( FriendlyEmail : AnsiString ) : AnsiString
12084:   Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString ) : AnsiString
12085:   Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString ) : AnsiString
12086:   Function AlGenerateInternetMessageID : AnsiString;
12087:   Function AlGenerateInternetMessageID1( ahostname : AnsiString ) : AnsiString;
12088:   Function ALDecodeQuotedPrintableString( src : AnsiString ) : AnsiString
12089:   Function AlDecodeInternetMessageHeaderInUTF8( aHeaderStr:AnsiString;aDefaultCodePage:Integer ):AnsiString;
12090: end;
12091:
12092: (*-----*)
12093: procedure SIRegister_ALFcWinSock(CL: TPSPascalCompiler);
12094: begin
12095:   Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12096:   Function ALIPAddrToName( IPAddr : AnsiString ) : AnsiString
12097:   Function ALGetLocalIPs : TALStrings
12098:   Function ALGetLocalHostName : AnsiString
12099: end;
12100:
12101: procedure SIRegister_ALFcncCGI(CL: TPSPascalCompiler);
12102: begin
12103:   Procedure A1CGIInitDefaultServerVariablesFromWebRequest(WebRequest :
TALWebRequest;ServerVariables:TALStrings);
12104:   Procedure A1CGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12105:   Procedure A1CGIInitDefaultServerVariables( ServerVariables : TALStrings );
12106:   Procedure A1CGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
ScriptFileName:AnsiString;Url:Ansistr;
12107:   Procedure A1CGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12108:   Procedure A1CGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
: Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12109:   Procedure A1CGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
WebRequest : TALIsapiRequest;
overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;
12110:   +'overloadedRequestContentStream:Tstream;var
ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12111:   Procedure A1CGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
ResponseHeader : TALHTTPResponseHeader);
12112: end;
12113:
12114: procedure SIRegister_ALFcExecute(CL: TPSPascalCompiler);
12115: begin
12116:   TStartupInfoA', 'TStartupInfo
12117:   'SE_CREATE_TOKEN_NAME', 'String'( 'SeCreateTokenPrivilege
12118:   SE_ASSIGNPRIMARYTOKEN_NAME', 'String 'SeAssignPrimaryTokenPrivilege
12119:   SE_LOCK_MEMORY_NAME', 'String( 'SeLockMemoryPrivilege
12120:   SE_INCREASE_QUOTA_NAME', 'String 'SeIncreaseQuotaPrivilege
12121:   SE_UNSOLICITED_INPUT_NAME', 'String 'SeUnsolicitedInputPrivilege
12122:   SE_MACHINE_ACCOUNT_NAME', 'String 'SeMachineAccountPrivilege
12123:   SE_TCB_NAME', 'String 'SeTcbPrivilege
12124:   SE_SECURITY_NAME', 'String 'SeSecurityPrivilege
12125:   SE_TAKE_OWNERSHIP_NAME', 'String 'SeTakeOwnershipPrivilege
12126:   SE_LOAD_DRIVER_NAME', 'String 'SeLoadDriverPrivilege
12127:   SE_SYSTEM_PROFILE_NAME', 'String 'SeSystemProfilePrivilege
12128:   SE_SYSTEMTIME_NAME', 'String 'SeSystemtimePrivilege
12129:   SE_PROF_SINGLE_PROCESS_NAME', 'String 'SeProfileSingleProcessPrivilege
12130:   SE_INC_BASE_PRIORITY_NAME', 'String 'SeIncreaseBasePriorityPrivilege
12131:   SE_CREATE_PAGEFILE_NAME', 'String 'SeCreatePagefilePrivilege
12132:   SE_CREATE_PERMANENT_NAME', 'String 'SeCreatePermanentPrivilege
12133:   SE_BACKUP_NAME', 'String 'SeBackupPrivilege
12134:   SE_RESTORE_NAME', 'String 'SeRestorePrivilege
12135:   SE_SHUTDOWN_NAME', 'String 'SeShutdownPrivilege
12136:   SE_DEBUG_NAME', 'String 'SeDebugPrivilege
12137:   SE_AUDIT_NAME', 'String 'SeAuditPrivilege
12138:   SE_SYSTEM_ENVIRONMENT_NAME', 'String 'SeSystemEnvironmentPrivilege
12139:   SE_CHANGE_NOTIFY_NAME', 'String 'SeChangeNotifyPrivilege
12140:   SE_REMOTE_SHUTDOWN_NAME', 'String 'SeRemoteShutdownPrivilege
12141:   SE_UNDOCK_NAME', 'String 'SeUndockPrivilege
12142:   SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege
12143:   SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege
12144:   SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege
12145:   Function AlGetEnvironmentString : AnsiString
12146:   Function ALWinExec32(const FileName,CurrentDir,
Environment:AnsiString;InStream:Tstream;OutStream:TStream) : Dword;

```

```

12147: Function ALWinExec321 (const FileName: AnsiString; InputStream:Tstream; OutputStream:TStream) : Dword;
12148: Function ALWinExecAndWait32 ( FileName : AnsiString; Visibility : integer ) : DWORD
12149: Function ALWinExecAndWait32V2 ( FileName : AnsiString; Visibility : integer ) : DWORD
12150: Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean ) : Boolean
12151: end;
12152:
12153: procedure SIRegister_ALFcns(CL: TPSPPascalCompiler);
12154: begin
12155:   Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
12156:     RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12157:   Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
12158:     RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime ) : Boolean;
12159:   Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
12160:     FileNameMask : ansiString; const FailIfExists : Boolean ) : Boolean
12161:   Function ALGetModuleName : ansistring
12162:   Function ALGetModuleFileNameWithoutExtension : ansistring
12163:   Function ALGetModulePath : ansistring
12164:   Function ALGetFileSize( const AFileName : ansistring ) : int64
12165:   Function ALGetFileVersion( const AFileName : ansistring ) : ansistring
12166:   Function ALGetFileCreationDate( const aFileName : Ansistring ) : TDateTime
12167:   Function ALGetFileLastWriteDate( const aFileName : Ansistring ) : TDateTime
12168:   Function ALGetFileLastAccessDate( const aFileName : Ansistring ) : TDateTime
12169:   Procedure ALSetFileCreationDate( const aFileName : Ansistring; const aCreationDate : TDateTime )
12170:   Function ALIsDirectoryEmpty( const directory : ansiString ) : boolean
12171:   Function ALExists( const Path : ansiString ) : boolean
12172:   Function ALDirectoryExists( const Directory : Ansistring ) : Boolean
12173:   Function ALCreateDir( const Dir : Ansistring ) : Boolean
12174:   Function ALRemoveDir( const Dir : Ansistring ) : Boolean
12175:   Function ALDeleteFile( const FileName : Ansistring ) : Boolean
12176:   Function ALRenameFile( const OldName, NewName : ansistring ) : Boolean
12177: end;
12178: procedure SIRegister_ALFcnsMime(CL: TPSPPascalCompiler);
12179: begin
12180:   NativeInt', 'Integer
12181:   NativeUInt', 'Cardinal
12182:   Function ALMimeBase64EncodeString( const S : AnsiString ) : AnsiString
12183:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString ) : AnsiString
12184:   Function ALMimeBase64DecodeString( const S : AnsiString ) : AnsiString
12185:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt ) : NativeInt
12186:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt ) : NativeInt
12187:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12188:     InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12189:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12190:     InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12191:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12192:     InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12193:   Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12194:     InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt ) : NativeInt;
12195:   Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
12196:     InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt );
12197:   Procedure ALMimeBase64DecodePartialEnd( const InputBuffer : TByteDynArray; OutputOffset : NativeInt; const
12198:     ByteBuffer : Cardinal; const ByteBufferSize : Cardinal ) : NativeInt;
12199:   Procedure ALMimeBase64Encode( const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
12200:     OutputBuf:TByteDynArray );
12201:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
12202:     OutputBuffer:TByteDynArray );
12203:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
12204:     OutputBuffer:TByteDynArray );
12205:   Function ALMimeBase64Decode( const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
12206:     OutputBuffer:TByteDynArray );
12207:   Procedure ALMimeBase64DecodePartialEnd( const InputBuffer : TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSize : Cardinal ) : NativeInt;
12208:   Procedure ALMimeBase64DecodePartialEnd1( out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
12209:     ByteBufferSize:Cardinal ) : NativeInt;
12210:   Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName )
12211:   Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName )
12212:   Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName )
12213:   Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream )
12214:   Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream:TStream;const OutputStream:TStream )
12215:   Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream )
12216:   'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76 );
12217:   'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3 );
12218:   'cALMimeBase64_BUFFER_SIZE','Longint'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4 );
12219:   Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings )
12220:   Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings )
12221:   Function ALGetDefaultFileExtFromMimeType( aContentType : AnsiString ) : AnsiString
12222:   Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString ) : AnsiString
12223: end;
12224: procedure SIRegister_ALXmlDoc(CL: TPSPPascalCompiler);
12225: begin
12226:   'cALXMLNodeMaxListSize','LongInt'( Maxint div 16 );
12227:   FindClass ('TOBJECT'),'TALXMLNode
12228:   FindClass ('TOBJECT'),'TALXMLNodeList
12229:   FindClass ('TOBJECT'),'TALXMLDocument
12230:   TALXMLParseProcessingInstructionEvent','Procedure ( Sender:TObject; const Target,Data:AnsiString )

```

```

12221: TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12222: TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12223: +'nst Name : AnsiString; const Attributes : TALStrings)
12224: TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12225: TALXMLNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12226: +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12227: +'ntDocType, ntDocFragment, ntNotation )
12228: TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12229: TALXMLDocOptions', 'set of TALXMLDocOption
12230: TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12231: TALXMLParseOptions', 'set of TALXMLParseOption
12232: TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12233: PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12234: SIRegister_EALXMLDocError(CL);
12235: SIRegister_TALXMLNodeList(CL);
12236: SIRegister_TALXMLNode(CL);
12237: SIRegister_TALXmlElementNode(CL);
12238: SIRegister_TALXmlAttributeNode(CL);
12239: SIRegister_TALXmlTextNode(CL);
12240: SIRegister_TALXmlDocumentNode(CL);
12241: SIRegister_TALXmlCommentNode(CL);
12242: SIRegister_TALXmlProcessingInstrNode(CL);
12243: SIRegister_TALXmlCDataNode(CL);
12244: SIRegister_TALXmlEntityRefNode(CL);
12245: SIRegister_TALXmlEntityNode(CL);
12246: SIRegister_TALXmlDocTypeNode(CL);
12247: SIRegister_TALXmlDocFragmentNode(CL);
12248: SIRegister_TALXmlNotificationNode(CL);
12249: SIRegister_TALXMLDocument(CL);
12250: cALXMLUTF8EncodingStr', 'String 'UTF-8
12251: cALXmlUTF8HeaderStr', 'String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12252: CALNSDelim', 'String' :
12253: CALXML', 'String 'xml
12254: CALVersion', 'String 'version
12255: CALEncoding', 'String 'encoding
12256: CALStandalone', 'String 'standalone
12257: CALDefaultNodeIndent', 'String '
12258: CALXmlDocument', 'String 'DOCUMENT
12259: Function ALCreateEmptyXMLDocument( const Rootname : AnsiString ) : TALXMLDocument
12260: Procedure ALClearXMLDocument( const rootname:AnsiString; xmldoc:TALXMLDocument;const
  EncodingStr:AnsiString );
12261: Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildnodeName,
  ChildnodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12262: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
  ChildnodeName, ChildnodeValue : AnsiString; const Recurse : Boolean ) : TalxmlNode
12263: Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
  Recurse: Boolean):TalxmlNode
12264: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
  AttributeName,AttributeValue : AnsiString; const Recurse : Boolean ) : TalxmlNode
12265: Function ALEExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString ) :
  AnsiString
12266: end;
12267:
12268: procedure SIRegister_TeCanvas(CL: TPPascalCompiler);
12269: //based on TEEProc, TeCanvas, TEEEngine, TChart
12270: begin
12271:   'TeePiStep','Double').setExtended( Pi / 180.0 );
12272:   'TeeDefaultPerspective','LongInt'( 100 );
12273:   'TeeMinAngle','LongInt'( 270 );
12274:   'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ) );
12275:   'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ) );
12276:   'teeclCream','LongWord( TColor ( $F0FBFF ) );
12277:   'teeclMedGray','LongWord').SetUInt( TColor ( $A4AOAO ) );
12278:   'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ) );
12279:   'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ) );
12280:   'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ) );
12281:   'teeclMedGray','LongWord').SetUInt( TColor ( $A4AOAO ) );
12282:   'TA_LEFT','LongInt'( 0 );
12283:   'TA_RIGHT','LongInt'( 2 );
12284:   'TA_CENTER','LongInt'( 6 );
12285:   'TA_TOP','LongInt'( 0 );
12286:   'TA_BOTTOM','LongInt'( 8 );
12287:   'teePATCOPY','LongInt'( 0 );
12288:   'NumCirclePoints','LongInt'( 64 );
12289:   'teeDEFAULT_CHARSET','LongInt'( 1 );
12290:   'teeANTIALIASED_QUALITY','LongInt'( 4 );
12291:   'TA_LEFT','LongInt'( 0 );
12292:   'bs_Solid','LongInt'( 0 );
12293:   'teepf24Bit','LongInt'( 0 );
12294:   'teepfDevice','LongInt'( 1 );
12295:   'CM_MOUSELEAVE','LongInt'( 10000 );
12296:   'CM_SYSCOLORCHANGE','LongInt'( 10001 );
12297:   'DC_BRUSH','LongInt'( 18 );
12298:   'DC_PEN','LongInt'( 19 );
12299:   teeCOLORREF', 'LongWord
12300:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12301: //TNotifyEvent', 'Procedure ( Sender : TObject)
12302: SIRegister_TFilterRegion(CL);
12303: SIRegister_IFormCreator(CL);

```

```

12304: SIRegister_TTeeFilter(CL);
12305: //TFilterClass', 'class of TTeeFilter
12306: SIRegister_TFilterItems(CL);
12307: SIRegister_TConvolveFilter(CL);
12308: SIRegister_TBlurFilter(CL);
12309: SIRegister_TTeePicture(CL);
12310: TPenEndStyle', '( esRound, esSquare, esFlat )
12311: SIRegister_TChartPen(CL);
12312: SIRegister_TChartHiddenPen(CL);
12313: SIRegister_TDottedGrayPen(CL);
12314: SIRegister_TDarkGrayPen(CL);
12315: SIRegister_TWhitePen(CL);
12316: SIRegister_TChartBrush(CL);
12317: TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12318: TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12319: SIRegister_TVview3DOptions(CL);
12320: FindClass('TOBJECT'), 'TTeeCanvas
12321: TTeeTransparency', 'Integer
12322: SIRegister_TTeeBlend(CL);
12323: FindClass('TOBJECT'), 'TCanvas3D
12324: SIRegister_TTeeShadow(CL);
12325: teeTGradientDirection', '(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
gdFromBottomLeft, gdRadial, gdDiagonalUp, gdDiagonalDown )
12326: FindClass('TOBJECT'), 'TSubGradient
12327: SIRegister_TCustomTeeGradient(CL);
12328: SIRegister_TSubGradient(CL);
12329: SIRegister_TTeeGradient(CL);
12330: SIRegister_TTeeFontGradient(CL);
12331: SIRegister_TTeeFont(CL);
12332: TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12333: TCanvasTextAlign', 'Integer
12334: TTeeCanvasHandle', 'HDC
12335: SIRegister_TTeeCanvas(CL);
12336: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12337: SIRegister_TFloatXYZ(CL);
12338: TPoint3D', 'record x : integer; y : integer; z : Integer; end
12339: TRGB', 'record blue: byte; green: byte; red: byte; end
12340: {TRGB=packed record
12341:   Blue : Byte;
12342:   Green : Byte;
12343:   Red : Byte;
12344:   //$/IFDEF CLX //Alpha : Byte; // Linux end;}
12345:
12346: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 :
12347:   +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12348: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12349: TCanvas3DPlane', '( cpX, cpY, cpZ )
12350: //IInterface', 'IUnknown
12351: SIRegister_TCanvas3D(CL);
12352: SIRegister_TTeeCanvas3D(CL);
12353: TTrianglePoints', 'Array[0..2] of TPoint;
12354: TFourPoints', 'Array[0..3] of TPoint;
12355: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12356: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12357: Function Point3D( const x, y, z : Integer) : TPoint3D
12358: Procedure SwapDouble( var a, b : Double)
12359: Procedure SwapInteger( var a, b : Integer)
12360: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12361: Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12362: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer) : TRect
12363: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12364: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12365: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12366: Procedure UnClipCanvas( ACanvas : TCanvas)
12367: Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12368: Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12369: Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12370: 'TeeCharForHeight','String 'W
12371: 'DarkerColorQuantity','Byte').SetUInt( 128 );
12372: 'DarkColorQuantity','Byte').SetUInt( 64 );
12373: TButtonGetColorProc', 'Function : TColor
12374: SIRegister_TTeeButton(CL);
12375: SIRegister_TButtonColor(CL);
12376: SIRegister_TComboFlat(CL);
12377: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12378: Function TeePoint( const ax, ay : Integer) : TPoint
12379: Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12380: Function PointInRect( const Rect : TRect; x, y : Integer) : Boolean;
12381: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12382: Function OrientRectangle( const R : TRect) : TRect
12383: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12384: Function PolygonBounds( const P : array of TPoint) : TRect
12385: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12386: Function RGBValue( const Color : TColor) : TRGB
12387: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12388: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12389: Function PointAtDistance( AFrom, ATo : TPoint; Adist : Integer) : TPoint
12390: Function TeeCull( const P : TFourPoints) : Boolean;
12391: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;

```

```

12392:   TSmootherStretchOption', '( ssBestQuality, ssBestPerformance )
12393: Procedure SmoothStretch( Src, Dst : TBitmap );
12394: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmootherStretchOption );
12395: Function TeeDistance( const x, y : Double ) : Double
12396: Function TeeLoadLibrary( const FileName : String ) : HInst
12397: Procedure TeeFreeLibrary( hLibModule : HMODULE )
12398: Procedure TeeBlendBitmaps( const Percent:Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint )
12399: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap )
12400: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12401:   SIRegister_ICanvasHyperlinks(CL);
12402:   SIRegister_ICanvasToolTips(CL);
12403: Function Supports( const Instance : IInterface; const IID : TGUID ) : Boolean
12404: end;
12405:
12406: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12407: begin
12408:   TOvcHdc', 'Integer
12409:   TOvcHWND', 'Cardinal
12410:   TOvcHdc', 'HDC
12411:   TOvcHWND', 'HWNDF
12412: Function LoadBaseBitmap( lpBitmapName : PChar ) : HBITMAP
12413: Function LoadBaseCursor( lpCursorName : PChar ) : HCURSOR
12414: Function ovCompStruct( const S1, S2, Size : Cardinal ) : Integer
12415: Function DefaultEpoch : Integer
12416: Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12417: Procedure FixRealPrim( P : PChar; DC : Char )
12418: Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer ) : string
12419: Function GetLeftButton : Byte
12420: Function GetNextDlgItem( Ctrl : TOvcHwnd ) : hWnd
12421: Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte )
12422: Function GetShiftFlags : Byte
12423: Function ovCreateRotatedFont( F : TFont; Angle : Integer ) : hFont
12424: Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12425: Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet ) : string
12426: Function ovIsForegroundTask : Boolean
12427: Function ovTrimLeft( const S : string ) : string
12428: Function ovTrimRight( const S : string ) : string
12429: Function ovQuotedStr( const S : string ) : string
12430: Function ovWordCount( const S : string; const WordDelims : TCharSet ) : Integer
12431: Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12432: Function PtrDiff( const P1, P2 : PChar ) : Word
12433: Procedure PtrInc( var P, Delta : Word )
12434: Procedure PtrDec( var P, Delta : Word )
12435: Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer )
12436: Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer )
12437: Function ovMinI( X, Y : Integer ) : Integer
12438: Function ovMaxI( X, Y : Integer ) : Integer
12439: Function ovMinL( X, Y : LongInt ) : LongInt
12440: Function ovMaxL( X, Y : LongInt ) : LongInt
12441: Function GenerateComponentName( PF : TWInControl; const Root : string ) : string
12442: Function PartialCompare( const S1, S2 : string ) : Boolean
12443: Function PathEllipsis( const S : string; MaxWidth : Integer ) : string
12444: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor ) : TBitmap
12445: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas )
12446: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
TransparentColor : TColor )
12447: Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width:Int;Rect:TRect;
TransparentColor : TColorRef)
12448: Function ovWidthOf( const R : TRect ) : Integer
12449: Function ovHeightOf( const R : TRect ) : Integer
12450: Procedure ovDebugOutput( const S : string )
12451: Function GetArrowWidth( Width, Height : Integer ) : Integer
12452: Procedure StripCharSeq( CharSeq : string; var Str : string )
12453: Procedure StripCharFromEnd( aChr : Char; var Str : string )
12454: Procedure StripCharFromFront( aChr : Char; var Str : string )
12455: Function SystemParametersInfo(uiAction,uiParam: UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12456: Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12457: Function SystemParametersInfoA(uiAction,uiParam: UINT; pvParam: UINT; fWinIni : UINT) : BOOL
12458: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer ) : HRGN
12459: Function CreateEllipticRgnIndirect( const p1 : TRect ) : HRGN
12460: Function CreateFontIndirect( const p1 : TLogFont ) : HFONT
12461: Function CreateMetaFile( p1 : PChar ) : HDC
12462: Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12463: Function DrawText(HDC: HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12464: Function DrawTextS(HDC: HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12465: Function SetMapperFlags( DC : HDC; Flag : DWORD ) : DWORD
12466: Function SetGraphicsMode( hdc : HDC; iMode : Integer ) : Integer
12467: Function SetMapMode( DC : HDC; p2 : Integer ) : Integer
12468: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar ) : HMETAFILE
12469: //Function SetPaletteEntries(Palette:HPalette;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12470: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF ) : COLORREF
12471: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF ) : BOOL
12472: //Function SetPixelFormat(DC:HDC;PixelFormat:Integer;FormatDef:PPixelFormatDescriptor) : BOOL
12473: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer ) : Integer
12474: Function StretchBit(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
SrcHeight:Int;Rop:DWORD):BOOL
12475: Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer ) : BOOL

```

```

12476: Function StretchDIBits( DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
12477:   SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12478: Function SetROP2( DC : HDC; p2 : Integer) : Integer
12479: Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12480: Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12480: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12481: Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12482: Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12483: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12484: Function UpdateColors( DC : HDC) : BOOL
12485: Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12486: Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12487: Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12488: Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12489: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12490: Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12491: Function MaskBlt(DestDC:HDC; XDest,YDest,Width:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
12491:   HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12492: Function PglBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
12492:   yMask:Int):BOOL;
12493: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12494: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12495: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12496: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12497: Function PlayMetafile( DC : HDC; MF : HMETAFILE) : BOOL
12498: Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12499: Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12500: Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12501: Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12502: Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12503: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12504: Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12505: end;
12506:
12507: procedure SIRegister_ovcfiler(CL: TPSPPascalCompiler);
12508: begin
12509:   SIRegister_TOvcAbstractStore(CL);
12510:   //PExPropInfo', '^TExPropInfo' // will not work
12511:   // _TEXPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12512:   SIRegister_TOvcPropertyList(CL);
12513:   SIRegister_TOvcDataFiler(CL);
12514:   Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12515:   Procedure UpdateStoredlist1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12516:   Function CreateStoredItem( const CompName, PropName : string) : string
12517:   Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12518:   //Function GetType( PropInfo : PExPropInfo) : PTypeInfo
12519: end;
12520:
12521: procedure SIRegister_ovccoco(CL: TPSPPascalCompiler);
12522: begin
12523:   'ovsetsize','LongInt'( 16 );
12524:   'etSyntax','LongInt'( 0 );
12525:   'etSemantic','LongInt'( 1 );
12526:   'chCR','Char #13';
12527:   'chLF','Char #10';
12528:   'chLineSeparator',' chCR');
12529:   SIRegister_TCocoError(CL);
12530:   SIRegister_TCommentItem(CL);
12531:   SIRegister_TCommentList(CL);
12532:   SIRegister_TSymbolPosition(CL);
12533:   TGenListType', '(
12533:     glNever, glAlways, glOnError )
12534:   TovbitSet', 'set of Integer
12535:   //PStartTable', '^TStartTable' // will not work
12536:   'TovCharSet', 'set of AnsiChar
12537:   TAAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12538:   TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12539:   TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12540:   TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12541:   TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12542:     +'osition; const Data : string; ErrorType : integer)
12543:   TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12544:   TGetCH', 'Function ( pos : longint ) : char
12545:   TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12546:   SIRegister_TCocoRScanner(CL);
12547:   SIRegister_TCocoRGrammar(CL);
12548:   '_EF','Char #0);
12549:   '_TAB','Char').SetString( #09);
12550:   '_CR','Char').SetString( #13);
12551:   '_LF','Char').SetString( #10);
12552:   '_EL','',').SetString( _CR);
12553:   '_EOF','Char').SetString( #26);
12554:   'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12555:   'minErrDist','LongInt'( 2 );
12556:   Function ovPadL( S : string; ch : char; L : integer) : string
12557: end;
12558:
12559:   TFormatSettings = record
12560:     CurrencyFormat: Byte;
12561:     NegCurrFormat: Byte;

```

```

12562:     ThousandSeparator: Char;
12563:     DecimalSeparator: Char;
12564:     CurrencyDecimals: Byte;
12565:     DateSeparator: Char;
12566:     TimeSeparator: Char;
12567:     ListSeparator: Char;
12568:     CurrencyString: string;
12569:     ShortDateFormat: string;
12570:     LongDateFormat: string;
12571:     TimeAMString: string;
12572:     TimePMString: string;
12573:     ShortTimeFormat: string;
12574:     LongTimeFormat: string;
12575:     ShortMonthNames: array[1..12] of string;
12576:     LongMonthNames: array[1..12] of string;
12577:     ShortDayNames: array[1..7] of string;
12578:     LongDayNames: array[1..7] of string;
12579:     TwoDigitYearCenturyWindow: Word;
12580:   end;
12581;
12582: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12583: begin
12584:   Function ovFormatSettings : TFormatSettings
12585:   end;
12586;
12587: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12588: begin
12589:   TOvcCharSet', 'set of Char
12590:   ovBTable', 'array[0..255] of Byte
12591: //BTable = array[0..{SIFDEF UNICODE}]{$IFDEF
12592:   Huge_UNICODE_BMTABLE}{$FFFF{$ELSE}{$ENDIF}{$ENDIF}} of Byte;
12593:   Function BinaryBPChar( Dest : PChar; B : Byte ) : PChar
12594:   Function BinaryLPChar( Dest : PChar; L : LongInt ) : PChar
12595:   Function BinaryWPChar( Dest : PChar; W : Word ) : PChar
12596:   Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable )
12597:   Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable;MatchString:PChar;var Pos:Cardinal):Bool;
12598:   Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean;
12599:   Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal ) : PChar
12600:   Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte ) : PChar
12601:   Function HexBPChar( Dest : PChar; B : Byte ) : PChar
12602:   Function HexLPChar( Dest : PChar; L : LongInt ) : PChar
12603:   Function HexPtrPChar( Dest : PChar; P : TObject ) : PChar
12604:   Function HexWPChar( Dest : PChar; W : Word ) : PChar
12605:   Function LoCaseChar( C : Char ) : Char
12606:   Function OctalLPChar( Dest : PChar; L : LongInt ) : PChar
12607:   Function StrChDeletePrim( P : PChar; Pos : Cardinal ) : PChar
12608:   Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal ) : PChar
12609:   Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word )
12610:   Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal ) : PChar
12611:   Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal ) : PChar
12612:   Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal ) : PChar
12613:   Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal ) : PChar
12614:   Function StrStPos( P, S : PChar; var Pos : Cardinal ) : Boolean
12615:   Function StrToLongPChar( S : PChar; var I : LongInt ) : Boolean
12616:   Procedure TrimAllSpacesPChar( P : PChar )
12617:   Function TrimEmbeddedZeros( const S : string ) : string
12618:   Procedure TrimEmbeddedZerosPChar( P : PChar )
12619:   Function TrimTrailPrimPChar( S : PChar ) : PChar
12620:   Function TrimTrailPChar( Dest, S : PChar ) : PChar
12621:   Function TrimTrailingZeros( const S : string ) : string
12622:   Procedure TrimTrailingZerosPChar( P : PChar )
12623:   Function UpCaseChar( C : Char ) : Char
12624:   Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet ) : Boolean
12625:   Function ovc32StringIsCurrentCodePage( const S : WideString ) : Boolean;
12626: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal ) : Boolean;
12627: end;
12628;
12629: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12630: begin
12631:   //PRaiseFrame', '^TRaiseFrame // will not work
12632:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : _Poin'
12633:   +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12634:   Procedure SafeCloseHandle( var Handle : THandle )
12635:   Procedure ExchangeInteger( X1, X2 : Integer )
12636:   Procedure FillInteger( const Buffer, Size, Value : Integer )
12637:   Function LongMulDiv( Multi, Mult2, Div1 : Longint ) : Longint
12638:   Function afCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean
12639:
12640:   FILENAME_ADVAPI32      = 'ADVAPI32.DLL';
12641:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12642:   function AbortSystemShutdown(lpMachineName: PKOLOChar): BOOL; stdcall;
12643:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLOChar;
12644:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12645:     SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12646:     const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12647:     var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12648:   function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLOChar;
12649:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;

```

```

12650:     SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12651:     AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12652:     ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12653:     var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12654: function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12655:     HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12656:     SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12657:     AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12658:     ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12659:     var GrantedAccess: DWORD; var AccessStatusList: DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12660: function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12661: function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12662: function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12663:     lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12664:     lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12665:     dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12666:     const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12667: function GetCurrentUser(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12668: function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12669:     pSecurityDescriptor: PSecurityDescriptor; nLength: DWORD; var lpnLengthNeeded: DWORD): BOOL; stdcall;
12670: function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12671: function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12672:     dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12673: function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12674:     dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12675: function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12676:     Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12677:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12678: function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12679:     Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12680:     var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12681: function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12682:     lpDisplayName: PKOLChar; var cb DisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12683: function LookupPrivilegeName(lpSystemName: PKOLChar;
12684:     var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12685: function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12686:     var lpLuid: TLargeInteger): BOOL; stdcall;
12687: function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12688:     HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12689: function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12690:     HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12691: function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12692:     ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12693:     ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12694:     var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12695:     var GenerateOnClose: BOOL): BOOL; stdcall;
12696: function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12697:     HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12698:     var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12699: function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12700: function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12701: function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12702:     ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12703: function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12704:     lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12705:     var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12706: function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12707:     var phkResult: HKEY): Longint; stdcall;
12708: function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12709:     var phkResult: HKEY): Longint; stdcall;
12710: function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12711:     Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12712:     lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12713:     lpdwDisposition: PDWORD): Longint; stdcall;
12714: function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12715: function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12716: function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12717:     var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12718:     lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12719: function RegEnumKey(hKey:HKEY;dwIndex:DWORD;lpName:PKOLChar;cbName:DWORD):Longint;stdcall;
12720: function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12721:     var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12722:     lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12723: function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12724: function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
12725: function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12726:     ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12727: function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12728:     lpcbClass: PDWORD; lpReserved: Pointer;
12729:     lpcbSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12730:     lpcb.MaxValueNameLen, lpcb.MaxValueLen, lpcbSecurityDescriptor: PDWORD;
12731:     lpftLastWriteTime: PFileTime): Longint; stdcall;
12732: function RegQueryMultipleValues(hKey: HKEY; var ValList;
12733:     NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12734: function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12735:     lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12736: function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12737:     lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12738: function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
```

```

12739:     lpNewFile: PKOLChar; lpOldFile: PKOLChar) : Longint; stdcall;
12740:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD) : Longint; stdcall;
12741:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12742:         lpSecurityAttributes: PSecurityAttributes) : Longint; stdcall;
12743:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12744:         dwType: DWORD; lpData: PKOLChar; cbData: DWORD) : Longint; stdcall;
12745:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12746:         Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD) : Longint; stdcall;
12747:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar) : Longint; stdcall;
12748:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12749:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12750:         dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12751:         dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12752:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12753:         pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12754:
12755:     Function wAddAtom( lpString : PKOLChar ) : ATOM
12756:     Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL ) : THandle
12757:     //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12758:     lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12759:     //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12760:     Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
12761:         lpString2 : PKOLChar; cchCount2 : Integer ) : Integer
12762:     Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL ) : BOOL
12763:     //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
12764:         TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12765:     Function w.CreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes ) : BOOL
12766:     Function w.CreateDirectoryEx(lpTemplateDirectory,
12767:         lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribs):BOOL;
12768:     Function wCreateEvent( lpEventAttribes:PSecurityAttrib;bManualReset,
12769:         bInitialState:BOOL;lpName:PKOLChar):THandle;
12770:     Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
12771:         PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12772:     Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
12773:         dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar ) : THandle
12774:     Function wCreateHardLink( lpFileName,
12775:         lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12776:     Function
12777:     CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle;
12778:     Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
12779:         nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes ) : THandle
12780:     //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
12781:         lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
12782:             Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
12783:             lpProcessInfo:TProcessInformation):BOOL
12784:     Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
12785:         Longint; lpName : PKOLChar ) : THandle
12786:     Function wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle;
12787:     Function wDeleteDosDevice( dwFlags : WORD; lpDeviceName, lpTargetPath : PKOLChar ) : BOOL
12788:     Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL ) : BOOL
12789:     //Function
12790:     wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12791:     //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12792:     //Function
12793:     wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;iParam:Longint):BOOL;
12794:     //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;iParam:Longint):BOOL;
12795:     //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12796:     //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12797:     //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12798:     Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD ) : DWORD
12799:     Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12800:     //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
12801:         dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12802:     Function wFindAtom( lpString : PKOLChar ) : ATOM
12803:     Function
12804:     wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12805:     Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12806:     //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFIndexInfoLevels; lpFindFileData :
12807:         Pointer; fSearchOp : TFIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12808:     Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12809:     Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar ) : HRSRC
12810:     Function wFindResourceEx(hModule: HMODULE; lpType,lpName:PKOLChar; wLanguage: Word) : HRSRC
12811:     Function
12812:     wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer;
12813:     //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
12814:         DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer ) : DWORD
12815:     Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12816:     Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12817:     Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12818:     Function wGetCommandLine : PKOLChar
12819:     //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD ) : DWORD
12820:     Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD ) : BOOL
12821:     Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD ) : DWORD
12822:     //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
12823:         PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer ) : Integer
12824:     Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12825:     //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
12826:         lpDateStr : PKOLChar; cchDate : Integer ) : Integer

```

```

12805: //Function wGetDefaultCommConfig(lpszName:PKOLChar;var lpCC:TCommConfig;var lpdwSize:DWORD):BOOL
12806: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12807: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12808: Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12809: Function wGetEnvironmentStrings : PKOLChar
12810: Function wGetEnvironmentVariable(lpName:PKOLChar; lpBuffer:PKOLChar; nSize: DWORD) : DWORD;
12811: Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12812: //Function
12813: wGetFileAttributesEx(lpFileName:PKOLChar,fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12814: Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12815: //Function wGetLocaleInfo(Locale:LCID; LCTYPE:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12816: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12817: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12818: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12819: //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt,
lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12820: Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Int;lpFileName:PKOLChar):UINT;
12821: Function
12822: wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;sSize:DWORD;pFileName:PKOLChar):DWORD;
12823: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
nSize:DWORD; lpFileName : PKOLChar) : DWORD
12824: Function wGetProfileInt( lpAppName : PKOLChar; nDefault : Integer) : UINT
12825: Function wGetProfileSection( lpAppName:PKOLChar;lpReturnedString:PKOLChar;nSize:DWORD): DWORD
12826: Function wGetProfileString( lpAppName,lpKeyName,
lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD);
12827: Function wGetShortPathName(lpszLongPath:PKOLChar;lpszShortPath:PKOLChar;cchBuffer:DWORD) : DWORD
12828: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12829: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
lpCharType):BOOL
12830: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12831: Function wGetTempFileName(lpPathName,lpPrefixString:PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar) : UINT
12832: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12833: //Function
12834: wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12835: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12836: Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
: DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12837: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12838: Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12839: Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12840: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12841: Function
12842: wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12843: Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12844: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12845: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12846: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12847: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12848: Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12849: Function wOpenFileMapping(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpName: PKOLChar):THandle
12850: Function wOpenMutex(dwDesiredAccess:DWORD;bInheritHandle:BOOL; lpName : PKOLChar) : THandle
12851: Function wOpenSemaphore( dwDesiredAccess:DWORD; bInheritHandle:BOOL;lpName:PKOLChar):THandle
12852: Procedure wOutputDebugString( lpOutputString : PKOLChar)
12853: //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
lpNumberOfEventsRead:DWORD):BOOL;
12854: Function wQueryDosDevice(lpDeviceName:PKOLChar;lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12855: //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12856: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12857: //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
lpNumOfEventsRead:DWORD):BOOL;
12858: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
: TCoord; var lpReadRegion : TSmallRect) : BOOL
12859: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12860: Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12861: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
lpClipRectangle:PSmallRect;dwDestinationOrigin:TCoord;var lpFill:TCharInfo):BOOL
12862: Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12863: Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12864: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12865: Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12866: //Function wSetDefaultCommConfig(lpszName: PKOLChar; lpCC:PCommConfig; dwSize: DWORD) : BOOL
12867: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12868: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12869: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCData : PKOLChar) : BOOL
12870: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12871: //Function wUpdateResource(hUpdate:THandle;lpType,
lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD) : BOOL

```

```

12872: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12873: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12874: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite
12875: :DWORD; var lpNumberOfCharsWritten: DWORD; lpReserved : Pointer) : BOOL
12876: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
12877: var lpNumberOfEventsWritten : DWORD) : BOOL
12878: //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
12879: TCoord; var lpWriteRegion : TSmallRect) : BOOL
12880: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
12881: DWWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12882: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12883: Function wWritePrivateProfileString(lpAppName,lpKeyName,lpString,lpFileName:PKOLChar) : BOOL
12884: Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12885: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12886: Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12887: Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12888: Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12889: Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12890: Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12891: Function wlstrlstrn( lpString : PKOLChar) : Integer
12892: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruct :
12893: PNetConnectInfoStruct) : DWORD
12894: //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
12895: lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12896: //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
12897: lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12898: Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12899: Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12900: Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12901: //Function wWNetConnectionDialog( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12902: //Function wWNetDisconnectDialog( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12903: //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12904: Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12905: Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
12906: : PKOLChar; nNameBufSize : DWORD) : DWORD
12907: //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12908: Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12909: //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12910: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
12911: lpBufferSize:DWORD):DWORD;
12912: Function wWNet GetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12913: //Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12914: //Function wWNetSetConnection(lpName:PKOLChar;dwProperties:DWORD; pvValues : Pointer) : DWORD
12915: //Function wWNetUseConnection(hwndOwner:HWND;var
12916: lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
12917: lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12918: Function wGetFileVersionInfo(lptstrFilenam:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12919: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12920: Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
12921: lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12922: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
12923: szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12924: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12925: //Func wGetPrivateProfileStruct(lpszSection,
12926: lpszKey:PKOLChar;lpStruct:Ptr;lpvruSizeStruct:UINT;szFile:PKOLChar):BOOL;
12927: //Func wWritePrivateProfileStruct(lpszSection,
12928: lpszKey:PKOLChar;lpStruct:Ptr;lpvruSizeStruct:UINT;szFile:PKOLChar):BOOL;
12929: Function wAddFontResource( FileName : PKOLChar) : Integer
12930: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12931: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12932: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12933: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12934: //Function wCreateDC(lpszDriver,lpszDevice,lpszOutput:PKOLChar; lpdvmInit: PDeviceMode) : HDC
12935: //Function wCreateEnhMetaFile(DC:HDC; FileName: PKOLChar; Rect: PRect; Desc : PKOLChar) : HDC
12936: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientalion, fnWeight : Integer; fdwItalic,
12937: fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
12938: fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12939: Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12940: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12941: // Function wCreateIC(lpszDriver,lpszDevice, lpszOutput:PKOLChar; lpdvmInit: PDeviceMode) : HDC
12942: Function wCreateMetaFile( p1 : PKOLChar) : HDC
12943: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12944: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
12945: pPort:PKOLChar;iIdx:Int;lpOut:PKOLChar;DevMod:PDeviceMode):Int;
12946: // Function wEnumFontFamilies(DC:HDC; p2 : PKOLChar; p3 : TTFNFontEnumProc; p4 : LPARAM) : BOOL
12947: //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TTFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL;
12948: //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12949: //Function wEnumICMPProfiles( DC : HDC; ICMPProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12950: //Function wExtTextOut(DC:HDC;X,
12951: Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12952: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12953: //Function wGetCharABCWidthsFloat(DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12954: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12955: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12956: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12957: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD) : DWORD
12958: Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12959: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12960: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD

```

```

12942: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
12943:   lpvBuffer : Pointer; const lpmat2 : TMat2 ) : DWORD
12944:   Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar ) : BOOL
12945:   // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD ) : BOOL
12946:   Function wGetMetafile( p1 : PKOLChar ) : HMETAFILE
12947:   // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer ) : Integer
12948:   //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer ) : UINT
12949:   Function wGetTextExtentPoint32( DC:HDC; Str:PKOLChar; Count:Integer; var Size : TSize) : BOOL
12950:   Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12951:   Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar ) : Integer
12952:   //Function wGetTextMetrics( DC : HDC; var TM : TTTextMetric ) : BOOL
12953:   Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer ) : BOOL
12954:   Function wRemoveFontResource( FileName : PKOLChar ) : BOOL
12955:   //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : BOOL
12956:   //Function wResetDC( DC : HDC; const InitData : TDeviceMode ) : HDC
12957:   Function wSetICMProfile( DC : HDC; Name : PKOLChar ) : BOOL
12958:   //Function wStartDoc( DC : HDC; const p2 : TDocInfo ) : Integer
12959:   Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer ) : BOOL
12960:   Function wUpdateICMRegKey( p1 : DWORD; p2 : PKOLChar; p4 : UINT ) : BOOL
12961:   Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD ) : BOOL
12962:   //Function wwglUseFontOutlines( p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat ):BOOL
12963:   Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12964:   Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer ) : BOOL
12965:   //Function
12966:   wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12967:   //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD ) : Longint
12968:   // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
12969:   dwFlags : DWORD; lParam : Pointer ) : Longint
12970:   Function wCharLower( lpsz : PKOLChar ) : PKOLchar
12971:   Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12972:   Function wCharNext( lpsz : PKOLChar ) : PKOLChar
12973:   //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12974:   Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar ) : PKOLChar
12975:   // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12976:   Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12977:   Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12978:   Function wCharUpper( lpsz : PKOLChar ) : PKOLchar
12979:   Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer ) : Integer
12980:   Function wCreateAcceleratorTable( var Accel, Count : Integer ) : HACCEL
12981:   //Function wCreateDesktop(lpszDesktop,
12982:   lpszDevice:PKOLChar;pDevMode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12983:   //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
12984:   HWND; lpDialogFunc : TFNDialogProc; dwInitParam : LPARAM ) : HWND
12985:   //Function wCreateDialogParam( hInstance : HINST; lpTemplate : PKOLChar; hWndParent : HWND;
12986:   lpDialogFunc : TFNDialogProc; dwInitParam : LPARAM ) : HWND
12987:   Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
12988:   hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12989:   //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar,dwStyle DWWORD;X,Y,
12990:   nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstace:HINST;lpParam:Pointer):HWND
12991:   //Function wCreateWindowStation(lpinsta:PKOLChar;dwReserv,
12992:   dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12993:   Function wDefDlgProc( hDl : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12994:   Function wDefFrameProc( hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12995:   Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam:LPARAM):LRESULT;
12996:   Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM):LRESULT
12997:   //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
12998:   : HWND; lpDialogFunc : TFNDialogProc; dwInitParam : LPARAM ) : Integer
12999:   //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
13000:   : TFNDialogProc; dwInitParam : LPARAM ) : Integer
13001:   Function wDispatchMessage( const lpMsg : TMsg ) : Longint
13002:   Function wDlgDirList( hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
13003:   nIDStaticPath:Integer;uFileType:UINT):Integer;
13004:   Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
13005:   nIDStaticPath:Int;uFiletype:UINT):Int;
13006:   Function wDlgDirSelectComboBoxEx( hDl : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
13007:   Function wDlgDirSelectEx( hDl : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer ) : BOOL
13008:   //FuncwDrawState(dc:HDC;brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:TPARA;wDat:WPARA;x,y,cx,
13009:   cy:Int;Flags:UINT):BOOL;
13010:   Function wDrawText(HDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
13011:   Function wFindWindow( lpClassName, lpWindowName : PKOLChar ) : HWND
13012:   Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar ) : HWND
13013:   //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
13014:   pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
13015:   // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass ) : BOOL
13016:   //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx ) : BOOL
13017:   Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer ) : Integer
13018:   Function wGetClipboardFormatName(format:UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
13019:   Function wGetDlgItemText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
13020:   Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
13021:   Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
13022:   //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
13023:   Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
13024:   Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT ) : BOOL
13025:   Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
13026:   //Function wGetTabbedTextExtent( hdc : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
13027:   lpnTabStopPositions ) : DWORD

```

```

13016: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
13017:   lpnLengthNeed:DWORD)BOOL;
13018:   Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
13019:   Function wGetWindowModuleFileName(hWnd:HWND;pszFileName:PKOLChar;cchFileNameMax:UINT) : UINT
13020:   Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
13021: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:L PARA;nCnt,X,Y,nWidt,
13022:   nHeight:Int):BOOL;
13023:   Function wInsertMenu(hMenu:HMENU; uPosition,uFlags, uIDNewItem:UINT; lpNewItem: PKOLChar) : BOOL
13024: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
13025:   Function wIsCharAlpha( ch : KOLChar ) : BOOL
13026:   Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
13027:   Function wIsCharLower( ch : KOLChar ) : BOOL
13028:   Function wIsCharUpper( ch : KOLChar ) : BOOL
13029:   Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
13030:   Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar ) : HBITMAP
13031:   Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar ) : HCURSOR
13032:   Function wLoadCursorFromFile( lpfileName : PKOLChar ) : HCURSOR
13033:   Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar ) : HICON
13034:   Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT) : THandle
13035:   Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
13036:   Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar ) : HMENU
13037: //Function wLoadMenuIndirect( lpMenuTemplate : Pointer ) : HMENU
13038:   Function wLoadString(hInstanc:HINST;uID : UINT; lpBuffer:PKOLChar;nBufferMax:Integer):Integer
13039:   Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
13040:   Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhk1 : HKL ) : UINT
13041:   Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
13042:   Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word) : Integer
13043: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL
13044:   Function wModifyMenu(hMnu: HMENU; uPosition,uFlags,uIDNewItem:UINT; lpNewItem:PKOLChar) : BOOL
13045: //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
13046: //Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
13047: //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
13048:   Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
13049:   Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD) : HDESK
13050:   Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
13051:   Function wPeekMessage(var lpMsg:TMsg;hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
13052:   Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13053:   Function wPostThreadMessage(idThread:DWORD;Msg:UINT; wParam : WPARAM; lParam:LPARAM) : BOOL
13054:   Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
13055: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
13056: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
13057:   Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
13058: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD) : HDEVNOTIFY
13059:   Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
13060:   Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
13061:   Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13062:   Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
13063: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
13064:   lpResultCallBack : TFNSendAsycProc; dwData : DWORD ) : BOOL
13065:   Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
13066:   Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
13067:   Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar ) : BOOL
13068: //Function wSetMenuItemInfo(pl: HMENU; p2: UINT; p3: BOOL; const p4: TMenuItemInfo ) : BOOL
13069:   Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle ) : BOOL
13070: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD) : BOOL
13071:   Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
13072:   Function wSetWindowText( hWnd : HWND; lpString : PKOLChar ) : BOOL
13073: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFnHookProc ) : HHOOK
13074: //Function wSetWindowsHookEx(idHook:Integer;lfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD) : HHOOK;
13075: // Function wSystemParametersInfo(uiAction,uiParam:UINT; pvParam:Pointer; fWinIni: UINT):BOOL
13076:   Function wTabbedTextOut(hdc: HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
13077:   lpnTabStopPositions,nTabOrigin:Int):Longint;
13078:   Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg ) : Integer
13079:   Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
13080:   Function wVKeyScan( ch : KOLChar ) : SHORT
13081:   Function wVKeyScanEx( ch : KOLChar; dwhk1 : HKL ) : SHORT
13082:   Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD ) : BOOL
13083:   Function wwsprintf( Output : PKOLChar; Format : PKOLChar ) : Integer
13084:   Function wwwsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list ) : Integer
13085: //TestDrive!
13086: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13087: 'PROC_CONVERTSIDTOSTRINGSSIDA','String').SetString( 'ConvertSidToStringSidA
13088:   Function GetDomainUserSids(const domainName:String;const userName:String; var foundDomain:String):String;
13089:   Function GetLocalUserSidStr( const UserName : string ) : string
13090:   Function getPid4user( const domain : string; const user : string; var pid : dword ) : boolean
13091:   Function Impersonate2User( const domain : string; const user : string ) : boolean
13092:   Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString ) : Boolean
13093:   Function KillProcessbyname( const exename : string; var found : integer ) : integer
13094:   Function getWinProcessList : TStringList
13095:   function WaitTilClose(hWnd: Integer): Integer;
13096:   function DoUserMsgs: Boolean;
13097:   function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13098: procedure ShowMsg(hParent: Integer; const Mess, Title: String); //modal but NOT blockable
13099: procedure DeleteMsgForm(Handle: Integer);

```

```

13100: procedure DisableForms;
13101: function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13102: end;
13103;
13104: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13105: begin
13106:   'AfMaxSyncSlots','LongInt'( 64 );
13107:   'AfSynchronizeTimeout','LongInt'( 2000 );
13108:   TAFSyncSlotID', 'DWORD
13109:   TAFSyncStatistics', 'record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13110:   TAFsafeSyncEvent', 'Procedure ( ID : TAFSyncSlotID )
13111:   TAFsafeDirectSyncEvent', 'Procedure
13112:   Function AfNewSyncSlot( const AEvent : TAFsafeSyncEvent ) : TAFSyncSlotID
13113:   Function AfReleaseSyncSlot( const ID : TAFSyncSlotID ) : Boolean
13114:   Function AfEnableSyncSlot( const ID : TAFSyncSlotID; Enable : Boolean ) : Boolean
13115:   Function AfValidateSyncSlot( const ID : TAFSyncSlotID ) : Boolean
13116:   Function AfSyncEvent( const ID : TAFSyncSlotID; Timeout : DWORD ) : Boolean
13117:   Function AfDirectSyncEvent( Event : TAFsafeDirectSyncEvent; Timeout : DWORD ) : Boolean
13118:   Function AfIsSyncMethod : Boolean
13119:   Function AfSyncWnd : HWnd
13120:   Function AfSyncStatistics : TAFSyncStatistics
13121:   Procedure AfClearSyncStatistics
13122: end;
13123;
13124: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13125: begin
13126:   'fBinary','LongWord')($00000001);
13127:   'fParity','LongWord')($00000002);
13128:   'fOutxCtsFlow','LongWord').SetUInt($00000004);
13129:   'fOutxDsrFlow','LongWord')($00000008);
13130:   'fDtrControl','LongWord')($00000030);
13131:   'fDtrControlDisable','LongWord')($00000000);
13132:   'fDtrControlEnable','LongWord')($00000010);
13133:   'fDtrControlHandshake','LongWord')($00000020);
13134:   'fDsrsensitivity','LongWord')($00000040);
13135:   'fTXContinueOnXoff','LongWord')($00000080);
13136:   'fOutX','LongWord')($00000100);
13137:   'fInX','LongWord')($00000200);
13138:   'fErrorChar','LongWord')($00000400);
13139:   'fNull','LongWord')($00000800);
13140:   'fRtsControl','LongWord')($00000300);
13141:   'fRtsControlDisable','LongWord')($00000000);
13142:   'fRtsControlEnable','LongWord')($00000100);
13143:   'fRtsControlHandshake','LongWord')($00002000);
13144:   'fRtsControlToggle','LongWord')($00000300);
13145:   'fAbortOnError','LongWord')($00004000);
13146:   'fDummy2','LongWord')($FFFF8000);
13147:   TafCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13148:   FindClass('TOBJECT'), 'EAFComPortCoreError
13149:   FindClass('TOBJECT'), 'TAFComPortCore
13150:   TafComPortCoreEvent', 'Procedure ( Sender : TAFComPortCore; Event'
13151:     + 'tKind : TAFCoreEvent; Data : DWORD)
13152:   SIRegister_TAFComPortCoreThread(CL);
13153:   SIRegister_TAFComPortEventThread(CL);
13154:   SIRegister_TAFComPortWriteThread(CL);
13155:   SIRegister_TAFComPortCore(CL);
13156:   Function FormatDeviceName( PortNumber : Integer ) : string
13157: end;
13158;
13159: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13160: begin
13161:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word ) : TStream
13162:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String ) : Boolean
13163:   SIRegister_TApplicationFileIO(CL);
13164:   TDataFileCapability', '( dfcRead, dfcWrite )
13165:   TDataFileCapabilities', 'set of TDataFileCapability
13166:   SIRegister_TDataFile(CL);
13167:   //TDataFileClass', 'class of TDataFile
13168:   Function ApplicationFileIODefined : Boolean
13169:   Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13170:   Function FileStreamExists(const fileName: String) : Boolean
13171:   //Procedure Register
13172: end;
13173;
13174: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13175: begin
13176:   TALFBXFieldType', 'uftUnknown, uftNumeric, uftChar, uftVarchar'
13177:   + ', uftCString, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecision'
13178:   + 'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13179:   TALFBXScale', 'Integer
13180:   FindClass('TOBJECT'), 'EALFBXConvertError
13181:   SIRegister_EALFBXError(CL);
13182:   SIRegister_EALFBXException(CL);
13183:   FindClass('TOBJECT'), 'EALFBXGFixError
13184:   FindClass('TOBJECT'), 'EALFBXDSQLError
13185:   FindClass('TOBJECT'), 'EALFBXDynError
13186:   FindClass('TOBJECT'), 'EALFBXGBakError
13187:   FindClass('TOBJECT'), 'EALFBXGSecError
13188:   FindClass('TOBJECT'), 'EALFBXLicenseError

```

```

13189: FindClass ('TOBJECT'), 'EALFBXGStatError
13190: //EALFBXExceptionClass', 'class of EALFBXError
13191: TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4',
13192: + '37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865,
13193: + 'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13194: + 'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,
13195: + ' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13196: + 'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13197: + '_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13198: + '8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )'
13199: TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13200: + 'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13201: + 'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13202: + 'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )'
13203: TALFBXTransParams', 'set of TALFBXTransParam
13204: Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString ) : TALFBXCharacterSet
13205: Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13206: Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13207: 'cALFBXMaxParamLength', 'LongInt'( 125 );
13208: TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )'
13209: TALFBXParamsFlags', 'set of TALFBXParamsFlag
13210: //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13211: //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13212: TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,
13213: + 'stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommis'
13214: + 't, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )'
13215: SIRegister_TALFBXSQLDA(CL);
13216: //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13217: SIRegister_TALFBXPoolStream(CL);
13218: //PALFBXBlobData', '^TALFBXBlobData // will not work
13219: TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13220: //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13221: //TALFBXArrayDesc', 'TISCArrayDesc
13222: //TALFBXBlobDesc', 'TISCBlobDesc
13223: //PALFBXArrayInfo', 'TALFBXArrayInfo // will not work
13224: //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13225: SIRegister_TALFBXSQLResult(CL);
13226: //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13227: SIRegister_TALFBXSQLParams(CL);
13228: //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13229: TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13230: + 'atementType : TALFBXStatementType; end
13231: FindClass ('TOBJECT'), 'TALFBXLibrary
13232: //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13233: TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary )
13234: //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13235: +' Excep : EALFBXExceptionClass )
13236: SIRegister_TALFBXLibrary(CL);
13237: 'cALFBXDateOffset', 'LongInt'( 15018 );
13238: 'cALFBXTimeCoef', 'LongInt'( 864000000 );
13239: //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double );
13240: //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp );
13241: //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp ) : Double;
13242: Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word );
13243: Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13244: //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp );
13245: //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp );
13246: //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp );
13247: Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer ) : Integer
13248: Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord ) : Cardinal
13249: TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgnr )
13250: TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13251: Function ALFBXSQLQuote( const name : AnsiString ) : AnsiString
13252: Function ALFBXSQLUnQuote( const name : AnsiString ) : AnsiString
13253: end;
13254:
13255: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13256: begin
13257: TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13258: TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13259: TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13260: + 'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13261: + 'teger; First : Integer; CacheThreshold : Integer; end
13262: TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13263: TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13264: TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13265: TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13266: +' writes : int64; page_fetcheds : int64; page_marks : int64; end
13267: SIRegister_TALFBXClient(CL);
13268: SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13269: SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13270: SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13271: SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13272: SIRegister_TALFBXReadTransactionPoolContainer(CL);
13273: SIRegister_TALFBXReadStatementPoolContainer(CL);
13274: SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13275: SIRegister_TALFBXConnectionPoolClient(CL);
13276: SIRegister_TALFBXEventThread(CL);
13277: Function AlMySqlClientSlashedStr( const Str : AnsiString ) : AnsiString

```

```

13278: end;
13279:
13280: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13281: begin
13282:   _OSVERSIONINFOA = record
13283:     dwOSVersionInfoSize: DWORD;
13284:     dwMajorVersion: DWORD;
13285:     dwMinorVersion: DWORD;
13286:     dwBuildNumber: DWORD;
13287:     dwPlatformId: DWORD;
13288:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13289:   end;
13290:   TOSVersionInfoA', '_OSVERSIONINFOA
13291:   TOSVersionInfo', '_TOSVersionInfo
13292:   'WS_EX_RIGHT','LongWord')($00001000);
13293:   'WS_EX_LEFT','LongWord')($00000000);
13294:   'WS_EX_RTLREADING','LongWord')($00002000);
13295:   'WS_EX_LTRREADING','LongWord')($00000000);
13296:   'WS_EX_LEFTSCROLLBAR','LongWord')($00004000);
13297:   'WS_EX_RIGHTSCROLLBAR','LongWord')($00000000);
13298:   Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13299:   'LAYOUTRTL','LongWord')($00000001);
13300:   'LAYOUTBTT','LongWord')($00000002);
13301:   'LAYOUTVBT','LongWord')($00000004);
13302:   'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')($00000008);
13303:   'NOMIRRORBITMAP','LongWord')(DWORD($80000000));
13304:   Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13305:   Function GetLayout( dc : hdc) : DWORD
13306:   Function IsBidi : Boolean
13307:   Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13308:   Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13309:   Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13310:   Function GetPriorityClass( hProcess : THandle) : DWORD
13311:   Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13312:   Function CloseClipboard : BOOL
13313:   Function GetClipboardSequenceNumber : DWORD
13314:   Function GetClipboardOwner : HWND
13315:   Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13316:   Function GetClipboardViewer : HWND
13317:   Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13318:   Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13319:   Function GetClipboardData( uFormat : UINT) : THandle
13320:   Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13321:   Function CountClipboardFormats : Integer
13322:   Function EnumClipboardFormats( format : UINT) : UINT
13323:   Function GetClipboardFormatName( format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13324:   Function EmptyClipboard : BOOL
13325:   Function IsClipboardFormatAvailable( format : UINT) : BOOL
13326:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13327:   Function GetOpenClipboardWindow : HWND
13328:   Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13329:   Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13330:   Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL) : BOOL
13331:   Function GetDlgItemInt( hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL) : UINT
13332:   Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13333:   Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13334:   Function CheckRadioButton( hDlg:HWND;nIDFirstButton,nIDLastButton,nIDCheckButton:Integer):BOOL
13335:   Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13336:   Function SendDlgItemMessage( hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13337: end;
13338:
13339: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13340: begin
13341:   Function glExecuteAndWait( cmdLine:String; visibility:Word; timeout:Cardinal; killAppOnTimeOut:Bool) : Int;
13342:   Function GetTemporaryFilesPath : String
13343:   Function GetTemporaryFileName : String
13344:   Function FindFileInPaths( const fileName, paths : String) : String
13345:   Function PathsToString( const paths : TStrings) : String
13346:   Procedure StringToPaths( const pathsString : String; paths : TStrings)
13347:   //Function MacroExpandPath( const aPath : String) : String
13348: end;
13349:
13350: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13351: begin
13352:   SIRegister_TALMultiPartBaseContent(CL);
13353:   SIRegister_TALMultiPartBaseContents(CL);
13354:   SIRegister_TALMultiPartBaseStream(CL);
13355:   SIRegister_TALMultipartBaseEncoder(CL);
13356:   SIRegister_TALMultipartBaseDecoder(CL);
13357:   Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13358:   Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13359:   Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13360: end;
13361:
13362: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13363: begin
13364:   TdriveSize', 'record FreeS : Int64; Totals : Int64; end
13365:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : Integer;
13366:   +teger; WinMinorVersion : Integer; WinBuildNumber : Integer; WinCSDVersion: String; end

```

```

13367: Function aAllocPadedMem( Size : Cardinal ) : TObject
13368: Procedure aFreePadedMem( var P : TObject );
13369: Procedure aFreePadedMem( var P : PChar );
13370: Function aCheckPadedMem( P : Pointer ) : Byte
13371: Function aGetPadMemSize( P : Pointer ) : Cardinal
13372: Function aAllocMem( Size : Cardinal ) : Pointer
13373: Function aStrLen( const Str : PChar ) : Cardinal
13374: Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal ) : PChar
13375: Function aStrECopy( Dest : PChar; const Source : PChar ) : PChar
13376: Function aStrCopy( Dest : PChar; const Source : PChar ) : PChar
13377: Function aStrEnd( const Str : PChar ) : PChar
13378: Function aStrScan( const Str : PChar; aChr : Char ) : PChar
13379: Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal ) : PChar
13380: Function aPCharLength( const Str : PChar ) : Cardinal
13381: Function aPCharUpper( Str : PChar ) : PChar
13382: Function aPCharLower( Str : PChar ) : PChar
13383: Function aStrCat( Dest : PChar; const Source : PChar ) : PChar
13384: Function aLastDelimiter( const Delimiters, S : String ) : Integer
13385: Function aCopyTail( const S : String; Len : Integer ) : String
13386: Function aInt2Thos( I : Int64 ) : String
13387: Function aUpperCase( const S : String ) : String
13388: Function aLowerCase( const S : string ) : String
13389: Function aCompareText( const S1, S2 : string ) : Integer
13390: Function aSameText( const S1, S2 : string ) : Boolean
13391: Function aInt2Str( Value : Int64 ) : String
13392: Function aStr2Int( const Value : String ) : Int64
13393: Function aStr2IntDef( const S : string; Default : Int64 ) : Int64
13394: Function aGetFileExt( const FileName : String ) : String
13395: Function aGetFilePath( const FileName : String ) : String
13396: Function aGetFileName( const FileName : String ) : String
13397: Function aChangeExt( const FileName, Extension : String ) : String
13398: Function aAdjustLineBreaks( const S : string ) : string
13399: Function aGetWindowStr( WinHandle : HWND ) : String
13400: Function aDiskSpace( Drive : String ) : TdriveSize
13401: Function aFileExists( FileName : String ) : Boolean
13402: Function aFileSize( FileName : String ) : Int64
13403: Function aDirectoryExists( const Name : string ) : Boolean
13404: Function aSysErrorMessage( ErrorCode : Integer ) : string
13405: Function aShortPathName( const LongName : string ) : string
13406: Function aGetWindowVer : TWinVerRec
13407: procedure InitDriveSpacePtr;
13408: end;
13409:
13410: procedure SIRegister_MakeApp(CL: TPSPPascalCompiler);
13411: begin
13412:   aZero', 'LongInt'( 0 );
13413:   'makeappDEF', 'LongInt'( - 1 );
13414:   'CS_VREDRAW', 'LongInt'( DWORD( 1 ) );
13415:   'CS_HREDRAW', 'LongInt'( DWORD( 2 ) );
13416:   'CS_KEYCUTWINDOW', 'LongInt'( 4 );
13417:   'CS_DBLCLKS', 'LongInt'( 8 );
13418:   'CS_OWNDC', 'LongWord'( $20 );
13419:   'CS_CLASSDC', 'LongWord'( $40 );
13420:   'CS_PARENTDC', 'LongWord'( $80 );
13421:   'CS_NOKEYCUT', 'LongWord'( $100 );
13422:   'CS_NOCLOSE', 'LongWord'( $200 );
13423:   'CS_SAVEBITS', 'LongWord'( $800 );
13424:   'CS_BYTEALIGNCLIENT', 'LongWord'( $1000 );
13425:   'CS_BYTEALIGNWINDOW', 'LongWord'( $2000 );
13426:   'CS_GLOBALCLASS', 'LongWord'( $4000 );
13427:   'CS_IME', 'LongWord'( $10000 );
13428:   'CS_DROPSHADOW', 'LongWord'( $20000 );
13429:   //TPanelFunc', '^TPanelFunc // will not work
13430:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13431:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13432:   TFontLooks', 'set of TFontLook
13433:   TMessagefunc', 'function(hWnd, iMsg, wParam, lParam: Integer): Integer)
13434:   Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13435:   Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13436:   Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer ) : Word
13437:   Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13438:   Procedure RunMsgLoop( Show : Boolean )
13439:   Function MakeFont(Height,Width:Integer;const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13440:   Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int;
13441:   Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13442:   Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13443:   Function MakePanel(Left,Top,Width,Height,
hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13444:   Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13445:   Function id4menu( a, b : Byte; c : Byte; d : Byte ) : Cardinal
13446:   Procedure DoInitMakeApp //set first to init formclasscontrol!
13447: end;
13448:
13449: procedure SIRegister_ScreenSaver(CL: TPSPPascalCompiler);
13450: begin
13451:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13452:   +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13453:   TScreenSaverOptions', 'set of TScreenSaverOption

```

```

13454:  'cDefaultScreenSaverOptions', 'LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
13455:  ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13456:  TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13457:  SIRегистre_TScreenSaver(CL);
13458:  //Procedure Register
13459:  Procedure SetScreenSaverPassword
13460: end;
13461: procedure SIRегистre_XCollection(CL: TPSPascalCompiler);
13462: begin
13463:  FindClass('TOBJECT'), 'TXCollection
13464:  SIRегистre_EFilerException(CL);
13465:  SIRегистre_TXCollectionItem(CL);
13466:  //TXCollectionItemClass', 'class of TXCollectionItem
13467:  SIRегистre_TXCollection(CL);
13468:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13469:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13470:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13471:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13472:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13473:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13474: end;
13475:
13476: procedure SIRегистre_XOpenGL(CL: TPSPascalCompiler);
13477: begin
13478:  'TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond, mtcmArbitrary);
13479:  Procedure xglMapTexCoordToNull
13480:  Procedure xglMapTexCoordToMain
13481:  Procedure xglMapTexCoordToSecond
13482:  Procedure xglMapTexCoordToDual
13483:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13484:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13485:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13486:  Procedure xglBeginUpdate
13487:  Procedure xglEndUpdate
13488:  Procedure xglPushState
13489:  Procedure xglPopState
13490:  Procedure xglForbidSecondTextureUnit
13491:  Procedure xglAllowSecondTextureUnit
13492:  Function xglGetBitWiseMapping : Cardinal
13493: end;
13494:
13495: procedure SIRегистre_VectorLists(CL: TPSPascalCompiler);
13496: begin
13497:  TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13498:  TBaseListOptions', 'set of TBaseListOption
13499:  SIRегистre_TBaseList(CL);
13500:  SIRегистre_TBaseVectorList(CL);
13501:  SIRегистre_TAffineVectorList(CL);
13502:  SIRегистre_TVectorList(CL);
13503:  SIRегистre_TTexPointList(CL);
13504:  SIRегистre_TXIntegerList(CL);
13505:  //PSingleArrayList', '^TSingleArrayList // will not work
13506:  SIRегистre_TSingleList(CL);
13507:  SIRегистre_TByteList(CL);
13508:  SIRегистre_TQuaternionList(CL);
13509:  Procedure QuickSortLists( startIndex,endIndex:Integer;refList:TSingleList; objList : TList);
13510:  Procedure QuickSortLists1( startIndex,endIndex : Integer;refList: TSingleList; objList: TBaseList);
13511:  Procedure FastQuickSortLists( startIndex,
13512: endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13513: end;
13514: procedure SIRегистre_MeshUtils(CL: TPSPascalCompiler);
13515: begin
13516:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13517:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13518:  Procedure ConvertStripToList2(const strip:TAffineVectorList,const
13519:  indices:TIntegerList;list:TAffineVectorList);
13520:  Procedure ConvertIndexedListToList( const data:TAffineVectlist;const
13521:  indices:TIntegerList;list:TAffineVectorList);
13522:  Function BuildVectorCountOptimizedIndices( const vertices:TAffineVectorList; const
13523:  normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13524:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13525:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13526:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13527:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13528:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13529:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13530:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13531:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13532:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13533:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
13534:  edgesTriangles : TIntegerList) : TIntegerList
13535:  Procedure WeldVertices(vertices:TAffineVectorList;indicesMap:TIntegerList;weldRadius: Single)
13536:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
13537:  TPersistentObjectList;
13538:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13539:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
13540:  TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)

```

```

13535: end;
13536:
13537: procedure SIRegister_JclSysUtils(CL: TPSPPascalCompiler);
13538: begin
13539:   Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13540:   Procedure FreeMemAndNil( var P : TObject)
13541:   Function PCharOrNil( const S : string) : PChar
13542:     SIRegister_TJclReferenceMemoryStream(CL);
13543:     FindClass('TOBJECT'), 'EJclVMTError'
13544:     (Function GetVirtualMethodCount( AClass : TClass) : Integer
13545:      Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13546:      Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13547:        PDYNAMICIndexList', '^TDYNAMICIndexList // will not work
13548:        PDYNAMICAddressList', '^TDYNAMICAddressList // will not work
13549:        Function GetDynamicMethodCount( AClass : TClass) : Integer
13550:        Function GetDynamicIndexList( AClass : TClass) : PDYNAMICIndexList
13551:        Function GetDynamicAddressList( AClass : TClass) : PDYNAMICAddressList
13552:        Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13553:        Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13554:        Function GetInitTable( AClass : TClass) : PTypeInfo
13555:        PFIELDEntry', '^TFieldEntry // will not work)
13556:      TFieldEntry', 'record Offset : Integer; IDX : Word; Name : ShortString; end
13557:      Function JIsClass( Address : Pointer) : Boolean
13558:      Function JIsObject( Address : Pointer) : Boolean
13559:      Function GetImplementorOfInterface( const I : IInterface) : TObject
13560:      TDigitCount', 'Integer
13561:      SIRegister_TJclNumericFormat(CL);
13562:      Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13563:      TTextHandler', 'Procedure ( const Text : string)
13564: // 'ABORT_EXIT_CODE', 'LongInt'( ERROR_CANCELLED 1223);
13565:      Function JExecute(const
CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Cardinal;
13566:      Function JExecute(const Commandline:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13567:      Function ReadKey : Char //to and from the DOS console !
13568:      TModuleHandle', 'HINST
13569:      //TModuleHandle', 'Pointer
13570:      'INVALID_MODULEHANDLE_VALUE', 'LongInt'( TModuleHandle ( 0 ));
13571:      Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13572:      Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13573:      Procedure UnloadModule( var Module : TModuleHandle)
13574:      Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13575:      Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13576:      Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13577:      Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13578:      FindClass('TOBJECT'), 'EJclConversionError
13579:      Function JStrToBoolean( const S : string) : Boolean
13580:      Function JBooleanToStr( B : Boolean) : string
13581:      Function JIntToBool( I : Integer) : Boolean
13582:      Function JBoolToInt( B : Boolean) : Integer
13583:      'ListSeparator','String ';
13584:      'ListSeparator1','String ';
13585:      Procedure ListAddItems( var List : string; const Separator, Items : string)
13586:      Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13587:      Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13588:      Procedure ListDeleteItem( var List : string; const Separator : string; const Index : Integer)
13589:      Function ListItemCount( const List, Separator : string) : Integer
13590:      Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13591:      Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13592:      Function ListItemIndex( const List, Separator, Item : string) : Integer
13593:      Function SystemToObjectInstance : LongWord
13594:      Function IsCompiledWithPackages : Boolean
13595:      Function JJclGUIDToString( const GUID : TGUID) : string
13596:      Function JJclStringToGUID( const S : string) : TGUID
13597:      SIRegister_TJclInffCriticalSection(CL);
13598:      SIRegister_TJclSimpleLog(CL);
13599:      Procedure InitSimpleLog( const ALogFileFileName : string)
13600:    end;
13601:
13602: procedure SIRegister_JclBorlandTools(CL: TPSPPascalCompiler);
13603: begin
13604:   FindClass('TOBJECT'), 'EJclBorRADException
13605:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13606:   TJclBorRADToolEdition', '( deOPEN, dePRO, desVR )
13607:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13608:   TJclBorRADToolPath', 'string
13609:   'SupportedDelphiVersions','LongInt'( 5 or 6 or 7 or 8 or 9 or 10 or 11);
13610:   'SupportedBCBVersions','LongInt'( 5 or 6 or 10 or 11);
13611:   'SupportedBDSVersions','LongInt'( 1 or 2 or 3 or 4 or 5);
13612:   BorRADToolRepositoryPagesSection','String 'Repository Pages
13613:   BorRADToolRepositoryDialogsPage','String 'Dialogs
13614:   BorRADToolRepositoryFormsPage','String 'Forms
13615:   BorRADToolRepositoryProjectsPage','String 'Projects
13616:   BorRADToolRepositoryDataModulesPage','String 'Data Modules
13617:   BorRADToolRepositoryObjectType','String 'Type
13618:   BorRADToolRepositoryFormTemplate','String 'FormTemplate
13619:   BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13620:   BorRADToolRepositoryObjectName','String 'Name
13621:   BorRADToolRepositoryObjectPage','String 'Page
13622:   BorRADToolRepositoryObjectIcon','String 'Icon

```

```

13623: BorRADToolRepositoryObjectDescr', 'String 'Description
13624: BorRADToolRepositoryObjectAuthor', 'String 'Author
13625: BorRADToolRepositoryObjectAncestor', 'String 'Ancestor
13626: BorRADToolRepositoryObjectDesigner', 'String 'Designer
13627: BorRADToolRepositoryDesignerDfm', 'String 'dfm
13628: BorRADToolRepositoryDesignerXfm', 'String 'x fm
13629: BorRADToolRepositoryObjectNewForm', 'String 'DefaultNewForm
13630: BorRADToolRepositoryObjectMainForm', 'String 'Default MainForm
13631: SourceExtensionDelphiPackage', 'String '.dpk
13632: SourceExtensionBCBPackage', 'String '.bpk
13633: SourceExtensionDelphiProject', 'String '.dpr
13634: SourceExtensionBCBProject', 'String '.bpr
13635: SourceExtensionBDSProject', 'String '.bdspj
13636: SourceExtensionDProject', 'String '.dproj
13637: BinaryExtensionPackage', 'String '.bpl
13638: BinaryExtensionLibrary', 'String '.dll
13639: BinaryExtensionExecutable', 'String '.exe
13640: CompilerExtensionDCP', 'String '.dcp
13641: CompilerExtensionBPI', 'String '.bpi
13642: CompilerExtensionLIB', 'String '.lib
13643: CompilerExtensionTDS', 'String '.tds
13644: CompilerExtensionMAP', 'String '.map
13645: CompilerExtensionDRC', 'String '.drc
13646: CompilerExtensionDEF', 'String '.def
13647: SourceExtensionCPP', 'String '.cpp
13648: SourceExtensionH', 'String '.h
13649: SourceExtensionPAS', 'String '.pas
13650: SourceExtensionDFM', 'String '.dfm
13651: SourceExtensionXFM', 'String '.xfm
13652: SourceDescriptionPAS', 'String 'Pascal source file
13653: SourceDescriptionCPP', 'String 'C++ source file
13654: DesignerVCL', 'String 'VCL
13655: DesignerCLX', 'String 'CLX
13656: ProjectTypePackage', 'String 'package
13657: ProjectTypeLibrary', 'String 'library
13658: ProjectTypeProgram', 'String 'program
13659: Personality32Bit', 'String '32 bit
13660: Personality64Bit', 'String '64 bit
13661: PersonalityDelphi', 'String 'Delphi
13662: PersonalityDelphiDotNet', 'String 'Delphi.net
13663: PersonalityBCB', 'String 'C++Builder
13664: PersonalityCSB', 'String 'C#Builder
13665: PersonalityVB', 'String 'Visual Basic
13666: PersonalityDesign', 'String 'Design
13667: PersonalityUnknown', 'String 'Unknown personality
13668: PersonalityBDS', 'String 'Borland Developer Studio
13669: DOFDirectoriesSection', 'String 'Directories
13670: DOFUnitOutputDirKey', 'String 'UnitOutputDir
13671: DOFSearchPathName', 'String 'SearchPath
13672: DOFConditionals', 'String 'Conditionals
13673: DOFLinkerSection', 'String 'Linker
13674: DOFPackagesKey', 'String 'Packages
13675: DOFCompilerSection', 'String 'Compiler
13676: DOFPackageNoLinkKey', 'String 'PackageNoLink
13677: DOFAdditionalSection', 'String 'Additional
13678: DOFOptionsKey', 'String 'Options
13679: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13680:   +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13681:   +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13682: TJclBorPersonality', 'set of TJclBorPersonality
13683: TJclBorDesigner', '( bdVCL, bdCLX )
13684: TJclBorDesigners', 'set of TJclBorDesigner
13685: TJclBorPlatform', '( bp32bit, bp64bit )
13686: FindClass('TOBJECT'), 'TJclBorRADToolInstall
13687: SIRegister_TJclBorRADToolInstallationObject(CL);
13688: SIRegister_TJclBorlandOpenHelp(CL);
13689: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13690: TJclHelp2Objects', 'set of TJclHelp2Object
13691: SIRegister_TJclHelp2Manager(CL);
13692: SIRegister_TJclBorRADToolIDETool(CL);
13693: SIRegister_TJclBorRADToolIDEPackages(CL);
13694: SIRegister_IJclCommandLineTool(CL);
13695: FindClass('TOBJECT'), 'EJclCommandLineToolError
13696: SIRegister_TJclCommandLineTool(CL);
13697: SIRegister_TJclBorlandCommandLineTool(CL);
13698: SIRegister_TJclBCC32(CL);
13699: SIRegister_TJclDCC32(CL);
13700: TJclDCC', 'TJclDCC32
13701: SIRegister_TJclBpr2Mak(CL);
13702: SIRegister_TJclBorlandMake(CL);
13703: SIRegister_TJclBorRADToolPalette(CL);
13704: SIRegister_TJclBorRADToolRepository(CL);
13705: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13706: TCommandLineTools', 'set of TCommandLineTool
13707: //TJclBorRADToolInstall
13708: SIRegister_TJclBorRADToolInstallation(CL);
13709: SIRegister_TJclBCBInstallation(CL);
13710: SIRegister_TJclDelphiInstallation(CL);
13711: SIRegister_TJclDCCIL(CL);

```

```

13712: SIRegister_TJclBDSInstallation(CL);
13713: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13714: SIRegister_TJclBorRADToolInstallations(CL);
13715: Function BPLFileName( const BPLPath, PackageFileName : string) : string
13716: Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13717: Function IsDelphiPackage( const FileName : string) : Boolean
13718: Function IsDelphiProject( const FileName : string) : Boolean
13719: Function IsBCBPackage( const FileName : string) : Boolean
13720: Function IsBCBProject( const FileName : string) : Boolean
13721: Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffix:PString);
13722: Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13723: Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13724: Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
Descript:PString
13725: function SamePath(const Path1, Path2: string): Boolean;
13726: end;
13727:
13728: procedure SIRegister_JclFileUtils_max(CL: TPSpascalCompiler);
13729: begin
13730:   'ERROR_NO_MORE_FILES','LongInt'( 18);
13731:   //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13732:   //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13733:   //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13734:   'LPathSeparator','String '/'
13735:   'LDirDelimiter','String '//'
13736:   'LDirSeparator','String ':'
13737:   'JXPathDevicePrefix','String '\\.\\
13738:   'JXPathSeparator','String '\
13739:   'JXDirDelimiter','String '\
13740:   'JXDirSeparator','String ';
13741:   'JXPathUncPrefix','String '\\
13742:   'faNormalFile','LongWord')($00000080);
13743:   //''faUnixSpecific',' faSymLink';
13744:   JXTCompactPath', '( cpCenter, cpEnd )
13745:   _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13746:   +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime : '
13747:   +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13748:   TWIn32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13749:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13750:
13751: Function jxPathAddSeparator( const Path : string) : string
13752: Function jxPathAddExtension( const Path, Extension : string) : string
13753: Function jxPathAppend( const Path, Append : string) : string
13754: Function jxPathBuildRoot( const Drive : Byte) : string
13755: Function jxPathCanonicalize( const Path : string) : string
13756: Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13757: Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Int;CmpFmt:TCompactPath):string
13758: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13759: Function jxPathExtractFileDirFixed( const S : string) : string
13760: Function jxPathExtractFileNameNoExt( const Path : string) : string
13761: Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13762: Function jxPathGetDepth( const Path : string) : Integer
13763: Function jxPathGetLongName( const Path : string) : string
13764: Function jxPathGetShortName( const Path : string) : string
13765: Function jxPathGetLongName( const Path : string) : string
13766: Function jxPathGetShortName( const Path : string) : string
13767: Function jxPathGetRelativePath( Origin, Destination : string) : string
13768: Function jxPathGetTempPath : string
13769: Function jxPathIsAbsolute( const Path : string) : Boolean
13770: Function jxPathIsChild( const Path, Base : string) : Boolean
13771: Function jxPathIsDiskDevice( const Path : string) : Boolean
13772: Function jxPathIsUNC( const Path : string) : Boolean
13773: Function jxPathRemoveSeparator( const Path : string) : string
13774: Function jxPathRemoveExtension( const Path : string) : string
13775: Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13776: Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13777: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13778: JxTFileListOptions', 'set of TFileListOption
13779: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13780: TFileHandler', 'Procedure ( const FileName : string)
13781: TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13782: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13783: //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
FileMatchFunc:TFileMatchFunc):Bool;
13784: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int): Bool
13785: Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13786: Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13787: Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13788: Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13789: Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13790: Procedure jxCreatEmptyFile( const FileName : string)
13791: Function jxCloseVolume( var Volume : THandle) : Boolean
13792: Function jxDeleteDirectory(const DirectoryName: string; MoveToRecycleBin : Boolean) : Boolean
13793: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13794: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean

```

```

13795: Function jxDelTree( const Path : string ) : Boolean
13796: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13797: Function jxDiskInDrive( Drive : Char ) : Boolean
13798: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean ) : Boolean
13799: Function jxFileCreateTemp( var Prefix : string ) : THandle
13800: Function jxFileBackup( const FileName : string; Move : Boolean ) : Boolean
13801: Function jxFileCopy( const ExistingFileName, NewFileName: string; ReplaceExisting: Boolean ) : Boolean
13802: Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
13803: Function jxFileExists( const FileName : string ) : Boolean
13804: Function jxFileMove( const ExistingFileName, NewFileName: string; ReplaceExisting: Boolean ) : Boolean
13805: Function jxFileRestore( const FileName : string ) : Boolean
13806: Function jxGetBackupFileName( const FileName : string ) : string
13807: Function jxIsBackupFileName( const FileName : string ) : Boolean
13808: Function jxFileGetDisplayName( const FileName : string ) : string
13809: Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean ) : string
13810: Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean ) : string
13811: Function jxFileGetSize( const FileName : string ) : Int64
13812: Function jxFileGetTempName( const Prefix : string ) : string
13813: Function jxFileGetType( const FileName : string ) : string
13814: Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13815: Function jxForceDirectories( Name : string ) : Boolean
13816: Function jxGetDirectorySize( const Path : string ) : Int64
13817: Function jxGetDriveTypeStr( const Drive : Char ) : string
13818: Function jxGetFileAgeCoherence( const FileName : string ) : Boolean
13819: Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer )
13820: Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer )
13821: Function jxGetFileInfo( const FileName : string; out FileInfo : TSearchRec ) : Boolean;
13822: Function jxGetFileInfo( const FileName : string ) : TSearchRec;
13823: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
ResolveSymLinks:Boolean):Integer
13824: Function jxGetFileLastWrite( const FName : string ) : TFileTime;
13825: Function jxGetLastWrite( const FName : string; out LocalTime : TDateTime ) : Boolean;
13826: Function jxGetFileLastAccess( const FName : string ) : TFileTime;
13827: Function jxGetFileLastAccess( const FName : string; out LocalTime : TDateTime ) : Boolean;
13828: Function jxGetFileCreation( const FName : string ) : TFileTime;
13829: Function jxGetFileCreation( const FName : string; out LocalTime : TDateTime ) : Boolean;
13830: Function jxGetFileLastWrite( const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool ):Bool;
13831: Function jxGetFileLastWrite( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool ): Bool;
13832: Function jxGetFileLastWrite( const FName : string; ResolveSymLinks : Boolean ) : Integer;
13833: Function jxGetFileLastAccess( const FName:string; out TimeStamp:Integer;ResolveSymLinks:Bool ):Bool;
13834: Function jxGetFileLastAccess( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool ):Bool;
13835: Function jxGetFileLastAccess( const FName:string; ResolveSymLinks:Boolean ) : Integer;
13836: Function jxGetFileLastAttrChange( const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool ): Bool;
13837: Function jxGetFileLastAttrChange( const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool ):Bool;
13838: Function jxGetFileLastAttrChange( const FName : string; ResolveSymLinks:Boolean ) : Integer;
13839: Function jxGetModulePath( const Module : HMODULE ) : string
13840: Function jxGetSizeOfFile( const FileName : string ) : Int64;
13841: Function jxGetSizeOfFile( const FileInfo : TSearchRec ) : Int64;
13842: Function jxGetSizeOfFile( Handle : THandle ) : Int64;
13843: Function jxGetStandardFileInfo( const FileName : string ) : TWin32FileInfoAttributeData
13844: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean ) : Boolean
13845: Function jxIsRootDirectory( const CanonicalName : string ) : Boolean
13846: Function jxLockVolume( const Volume : string; var Handle : THandle ) : Boolean
13847: Function jxOpenVolume( const Drive : Char ) : THandle
13848: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime ) : Boolean
13849: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime ) : Boolean
13850: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime ) : Boolean
13851: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean
13852: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean
13853: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean
13854: Procedure jxShredFile( const FileName : string; Times : Integer )
13855: Function jxUnlockVolume( var Handle : THandle ) : Boolean
13856: Function jxCreatSymbolicLink( const Name, Target : string ) : Boolean
13857: Function jxSymbolicLinkTarget( const Name : string ) : string
13858: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13859: SIRegister_TJclCustomFileAttrMask(CL);
13860: SIRegister_TJclFileAttributeMask(CL);
13861: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13862: +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13863: TFileSearchOptions', 'set of TFileSearchOption
13864: TFileSearchTaskID', 'Integer
13865: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13866: +'hTaskID; const Aborted : Boolean)
13867: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13868: SIRegister_IJclFileEnumerator(CL);
13869: SIRegister_TJclFileEnumerator(CL);
13870: Function JxFileSearch : TJclfileEnumerator
13871: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )
13872: JxTFileFlags', 'set of TFileFlag
13873: FindClass('TOBJECT'), 'EJclFileVersionInfoError
13874: SIRegister_TJclFileVersionInfo(CL);
13875: Function jxOSIDToString( const OSID : DWORD ) : string
13876: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string
13877: Function jxVersionResourceAvailable( const FileName : string ) : Boolean
13878: TFileVersionFormat', '( vfMajorMinor, vfFull )
13879: Function jxFormatVersionString( const HiV, LoV : Word ) : string;
13880: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
13881: //Function FormatVersionString2(const FileInfo:TVSFixedFileInfo;VersionFrmat:TFileVersionFormat):str;
13882: //Procedure VersionExtractFileInfo(const FileInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);

```

```

13883: //Procedure VersionExtractProductInfo(const FileInfo:TVSFixedFileInfo;var Major,Minor,Build,
13884: Revision:Word);
13885: Function jxVersionFixedFileInfo( const FileName : string; var FileInfo : TVSFixedFileInfo ) : Boolean
13886: NotAvailableText : string;
13887: SIRegister_TJclTempFileStream(CL);
13888: FindClass('TOBJECT','TJclCustomFileMapping'
13889: SIRegister_TJclFileMappingView(CL);
13890: TJclFileMappingRoundOffset', '( rvDown, rvUp )
13891: SIRegister_TJclCustomFileMapping(CL);
13892: SIRegister_TJclSwapFileMapping(CL);
13893: SIRegister_TJclFileMappingStream(CL);
13894: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13895: //PPCharArray', '^TPCharArray // will not work
13896: SIRegister_TJclMappedTextReader(CL);
13897: SIRegister_TJclFileMaskComparator(CL);
13898: FindClass('TOBJECT','EJclPathError
13899: FindClass('TOBJECT','EJclFileUtilsError
13900: FindClass('TOBJECT','EJclTempFileStreamError
13901: FindClass('TOBJECT','EJclTempFileStreamError
13902: FindClass('TOBJECT','EJclFileMappingError
13903: FindClass('TOBJECT','EJclFileMappingViewError
13904: Function jxPathGetLongName2( const Path : string) : string
13905: Function jxWin32Deletefile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13906: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13907: Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13908: Function jxWin32RestoreFile( const FileName : string) : Boolean
13909: Function jxSamePath( const Path1, Path2 : string) : Boolean
13910: Procedure jxPathListAddItems( var List : string; const Items : string)
13911: Procedure jxPathListIncludeItems( var List : string; const Items : string)
13912: Procedure jxPathListDelItems( var List : string; const Items : string)
13913: Procedure jxPathListDelItem( var List : string; const Index : Integer)
13914: Function jxPathListItemCount( const List : string) : Integer
13915: Function jxPathListGetItem( const List : string; const Index : Integer) : string
13916: Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13917: Function jxPathListItemIndex( const List, Item : string) : Integer
13918: Function jxParamName(Idx:Int,const Separator:string,const AllowedPrefixChars:string;TrimName:Bool):string
13919: Function jxParamValue(Index:Integer, const Separator : string; TrimValue : Boolean) : string;
13920: Function jxParamValue1(const SearchName:string, const Separator : string; TrimValue : Boolean) : string;
13921: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
13922: AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13923: end;
13924: procedure SIRegister_FileUtil(CL: TFPSPascalCompiler);
13925: begin
13926: 'UTF8FileHeader','String #$ef#$bb#$bf';
13927: Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13928: Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13929: Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13930: Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLiks:boolean):int;
13931: Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13932: Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13933: Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13934: Procedure lCheckIfFileIsExecutable( const Afilename : string)
13935: Procedure lCheckIfFileIsSymlink( const Afilename : string)
13936: Function lFileIsReadable( const Afilename : string) : boolean
13937: Function lFileIsWritable( const Afilename : string) : boolean
13938: Function lFileIsText( const Afilename : string) : boolean
13939: Function lFileIsText( const Afilename : string; out FileReadable : boolean) : boolean
13940: Function lFileIsExecutable( const Afilename : string) : boolean
13941: Function lFileIsSymlink( const Afilename : string) : boolean
13942: Function lFileIsHardLink( const Afilename : string) : boolean
13943: Function lFileSize( const Filename : string) : int64;
13944: Function lGetFileDescription( const Afilename : string) : string
13945: Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13946: Function lTryReadallLinks( const Filename : string) : string
13947: Function lDirPathExists( const FileName : String) : Boolean
13948: Function lForceDirectory( DirectoryName : string) : boolean
13949: Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13950: Function lProgramDirectory : string
13951: Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13952: Function lExtractFileNameOnly( const Afilename : string) : string
13953: Function lExtractFileNameWithoutExt( const Afilename : string) : string
13954: Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13955: Function lCompareFileExt( const Filename, Ext : string) : integer;
13956: Function lFilenameIsPascalUnit( const Filename : string) : boolean
13957: Function lAppendPathDelim( const Path : string) : string
13958: Function lChompPathDelim( const Path : string) : string
13959: Function lTrimFilename( const Afilename : string) : string
13960: Function lCleanAndExpandFilename( const Filename : string) : string
13961: Function lCleanAndExpandDirectory( const Filename : string) : string
13962: Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13963: Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
13964: AlwaysRequireSharedBaseFolder : Boolean) : string
13965: Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13966: Function lFileIsInPath( const Filename, Path : string) : boolean
13967: Function lFileIsInDirectory( const Filename, Directory : string) : boolean

```

```

13967: TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13968: TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13969: 'AllDirectoryEntriesMask','String '*
13970: Function l GetAllFilesMask : string
13971: Function lGetExeExt : string
13972: Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags : TSearchFileInPathFlags) : string
13973: Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags : TSearchFileInPathFlags) : TString
13974: Function lFindDiskFilename( const Filename : string) : string
13975: Function lFindDiskFileNameInsensitive( const Filename : string) : string
13976: Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13977: Function lGetDarwinSystemFilename( Filename : string) : string
13978: SIRegister_TFileIterator(CL);
13979: TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13980: TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13981: TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13982: SIRegister_TFileSearcher(CL);
13983: Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13984: Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13985: // TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13986: // TCopyFileFlags', 'set of TCopyFileFlag
13987: Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13988: Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13989: Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13990: Function lReadFileToString( const Filename : string) : string
13991: Function lGetTempfilename( const Directory, Prefix : string) : string
13992: {Function NeedRTLAnsi : boolean
13993: Procedure SetNeedRTLAnsi( NewValue : boolean)
13994: Function UTF8ToSys( const s : string) : string
13995: Function SysToUTF8( const s : string) : string
13996: Function ConsoleToUTF8( const s : string) : string
13997: Function UTF8ToConsole( const s : string) : string
13998: Function FileExistsUTF8( const FileName : string) : boolean
13999: Function FileAgeUTF8( const FileName : string) : Longint
14000: Function DirectoryExistsUTF8( const Directory : string) : Boolean
14001: Function ExpandFileNameUTF8( const FileName : string) : string
14002: Function ExpandUNCFileNameUTF8( const FileName : string) : string
14003: Function ExtractShortPathNameUTF8( const FileName : String) : String
14004: Function FindFirstUTF8( const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
14005: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
14006: Procedure FindCloseUTF8( var F : TSearchrec)
14007: Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
14008: Function FileGetAttrUTF8( const FileName : String) : Longint
14009: Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
14010: Function DeleteFileUTF8( const FileName : String) : Boolean
14011: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
14012: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
14013: Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
14014: Function GetCurrentDirUTF8 : String
14015: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
14016: Function CreateDirUTF8( const NewDir : String) : Boolean
14017: Function RemoveDirUTF8( const Dir : String) : Boolean
14018: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
14019: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
14020: Function FileCreateUTF8( const FileName : string) : THandle;
14021: Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
14022: Function ParamStrUTF8( Param : Integer) : string
14023: Function GetEnvironmentStringUTF8( Index : Integer) : string
14024: Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
14025: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
14026: Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
14027: Function SysErrorMessageUTF8( ErrorCode : Integer) : String
14028: end;
14029:
14030: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
14031: begin
14032: //VK_F23 = 134;
14033: //{$EXTERNALSYM VK_F24}
14034: //VK_F24 = 135;
14035: TVirtualKeyCode', 'Integer
14036: 'VK_MOUSEWHEELUP','integer'(134);
14037: 'VK_MOUSEWHEELDOWN','integer'(135);
14038: Function glIsKeyDown( c : Char) : Boolean;
14039: Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
14040: Function glKeyPressed( minVkeyCode : TVirtualKeyCode) : TVirtualKeyCode
14041: Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
14042: Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
14043: Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
14044: Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
14045: end;
14046:
14047: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14048: begin
14049: TGLPoint', 'TPoint
14050: //PGLPoint', '^TGLPoint // will not work
14051: TGLRect', 'TRect
14052: //PGLRect', '^TGLRect // will not work
14053: TDelphiColor', 'TColor

```

```

14054: TGLPicture', 'TPicture
14055: TGLGraphic', 'TGraphic
14056: TGLBitmap', 'TBitmap
14057: //TGraphicClass', 'class of TGraphic
14058: TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14059: TGLMouseEvent', '( mbLeft, mbRight, mbMiddle )
14060: TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14061: +'Button; Shift : TShiftState; X, Y : Integer)
14062: TGLMouseEvent', 'TMouseEvent
14063: TGLKeyEvent', 'TKeyEvent
14064: TGLKeyPressEvent', 'TKeyPressEvent
14065: EGLOSError', 'EWin32Error
14066: EGLOSError', 'EWin32Error
14067: EGLOSError', 'EOSError
14068: 'glsAllFilter', 'string'All // sAllFilter
14069: Function GLPoint( const x, y : Integer) : TGLPoint
14070: Function GLRGB( const r, g, b : Byte) : TColor
14071: Function GLColorToRGB( color : TColor) : TColor
14072: Function GLGetRValue( rgb : DWORD) : Byte
14073: Function GLGetGValue( rgb : DWORD) : Byte
14074: Function GLGetBValue( rgb : DWORD) : Byte
14075: Procedure GLInitWinColors
14076: Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
14077: Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
14078: Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14079: Procedure GLInformationDlg( const msg : String)
14080: Function GLQuestionDlg( const msg : String) : Boolean
14081: Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14082: Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14083: Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14084: Function GLApplicationTerminated : Boolean
14085: Procedure GLRaiseLastOSError
14086: Procedure GLFreeAndNil( var anObject: TObject)
14087: Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14088: Function GLGetCurrentColorDepth : Integer
14089: Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14090: Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14091: Procedure GLSleep( length : Cardinal)
14092: Procedure GLQueryPerformanceCounter( var val : Int64)
14093: Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14094: Function GLStartPrecisionTimer : Int64
14095: Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14096: Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14097: Function GLRTDSC : Int64
14098: Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14099: Function GLOKMessageBox( const Text, Caption : string) : Integer
14100: Procedure GLShowHTMLUrl( Url : String)
14101: Procedure GLShowCursor( AShow : boolean)
14102: Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14103: Procedure GLGetCursorPos( var point : TGLPoint)
14104: Function GLGetScreenWidth : integer
14105: Function GLGetScreenHeight : integer
14106: Function GLGetTickCount : int64
14107: function RemoveSpaces(const str : String) : String;
14108: TNormalMapSpace', '( nmsObject, nmsTangent )
14109: Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14110: Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14111: Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
TAffineVectorList; Colors : TVectorList)
14112: Function CreateObjectSpaceNormalMap(Width,Height:Integer;HiNormals,
HiTexCoords:TAffineVectorList):TGLBitmap
14113: Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14114: end;
14115:
14116: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14117: begin
14118: TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14119: // PGLStarRecord', '^TGLStarRecord // will not work
14120: Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14121: Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14122: Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14123: end;
14124:
14125:
14126: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14127: begin
14128: TAABB', 'record min : TAffineVector; max : TAffineVector; end
14129: //PAABB', '^TAABB // will not work
14130: TBSphere', 'record Center : TAffineVector; Radius : single; end
14131: TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14132: TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14133: Function AddBB( var cl : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14134: Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14135: Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14136: Procedure SetAABB( var bb : TAABB; const v : TVector)
14137: Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14138: Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14139: Procedure AABBSScale( var bb : TAABB; const v : TAffineVector)

```

```

14140: Function BBMinX( const c : THmgBoundingBox ) : Single
14141: Function BBMaxX( const c : THmgBoundingBox ) : Single
14142: Function BBMinY( const c : THmgBoundingBox ) : Single
14143: Function BBMaxY( const c : THmgBoundingBox ) : Single
14144: Function BBMinZ( const c : THmgBoundingBox ) : Single
14145: Function BBMaxZ( const c : THmgBoundingBox ) : Single
14146: Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14147: Procedure AABBFromSweep(var SweepAABB : TAABB; const Start, Dest:TVector; const Radius : Single)
14148: Function AABBIntersection( const aabb1, aabb2 : TAABB ) : TAABB
14149: Function BBTxAABB( const aabb : THmgBoundingBox ) : TAABB
14150: Function AABBTxBB( const anAABB : TAABB ) : THmgBoundingBox;
14151: Function AABBTxBB1( const anAABB : TAABB; const m : TMatrix ) : THmgBoundingBox;
14152: Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14153: Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14154: Function IntersectAABBS( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix ) : Boolean;
14155: Function IntersectAABBSAbsoluteXY( const aabb1, aabb2 : TAABB ) : Boolean
14156: Function IntersectAABBSAbsoluteXZ( const aabb1, aabb2 : TAABB ) : Boolean
14157: Function IntersectAABBSAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14158: Function AABBfitsInAABBAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14159: Function PointInAABB( const p : TAffineVector; const aabb : TAABB ) : Boolean;
14160: Function PointInAABB1( const p : TVector; const aabb : TAABB ) : Boolean;
14161: Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB ) : boolean
14162: Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector ) : boolean
14163: Procedure ExtractAABCorners( const AABB : TAABB; var AABCorners : TAABCorners)
14164: Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14165: Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14166: Function BSphereToAABB1( const center : TAffineVector; radius : Single ) : TAABB;
14167: Function BSphereToAABB2( const center : TVector; radius : Single ) : TAABB;
14168: Function AABBContainsAABB( const mainAABB, testAABB : TAABB ) : TSpaceContains
14169: Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB ) : TSpaceContains
14170: Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere ) : TSpaceContains
14171: Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere ) : TSpaceContains
14172: Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
testBSphere:TBSphere):TSpaceContains
14173: Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere ) : TSpaceContains
14174: Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB ) : TSpaceContains
14175: Function ClipToAABB( const v : TAffineVector; const AABB : TAABB ) : TAffineVector
14176: Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere ) : boolean
14177: Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14178: Function AABBToclipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
viewportSizeY:Int):TClipRect
14179: end;
14180:
14181: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14182: begin
14183: Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14184: Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14185: Procedure Cylindrical_Cartesian2( const r, theta, z1 : single; var x, y, z : single; var ierr : integer);
14186: Procedure Cylindrical_Cartesian3( const r, theta, z1 : double; var x, y, z : double; var ierr : integer);
14187: Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14188: Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14189: Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14190: Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14191: Procedure Spherical_Cartesian2( const r, theta, phi : single; var x, y, z : single; var ierr : integer);
14192: Procedure Spherical_Cartesian3( const r, theta, phi : double; var x, y, z : double; var ierr : integer);
14193: Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14194: Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14195: Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14196: Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14197: Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14198: Procedure ProlateSpheroidal_Cartesian2( const xi, eta, phi, a : single; var x, y, z : single; var ierr : integer);
14199: Procedure ProlateSpheroidal_Cartesian3( const xi, eta, phi, a : double; var x, y, z : double; var ierr : integer);
14200: Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14201: Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14202: Procedure OblateSpheroidal_Cartesian2( const xi, eta, phi, a : single; var x, y, z : single; var ierr : integer);
14203: Procedure OblateSpheroidal_Cartesian3( const xi, eta, phi, a : double; var x, y, z : double; var ierr : integer);
14204: Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14205: Procedure BipolarCylindrical_Cartesian1( const u, v, z1, a : double; var x, y, z : double);
14206: Procedure BipolarCylindrical_Cartesian2( const u, v, z1, a : single; var x, y, z : single; var ierr : integer);
14207: Procedure BipolarCylindrical_Cartesian3( const u, v, z1, a : double; var x, y, z : double; var ierr : integer);
14208: end;
14209:
14210: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14211: begin
14212: 'EPSILON','Single').setExtended( 1e-40);
14213: 'EPSILON2','Single').setExtended( 1e-30); }
14214: TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14215: +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14216: THmgPlane', 'TVector
14217: TDoubleHmgPlane', 'THomogeneousDblVector
14218: {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14219: +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14220: +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW );
14221: TSingleArray', 'array of Single
14222: TTransformations', 'array [0..15] of Single)
14223: TPackedRotationMatrix', 'array [0..2] of Smallint)
14224: TVertex', 'TAffineVector
14225: //TVectorGL', 'THomogeneousFltVector
14226: //TMatrixGL', 'THomogeneousFltMatrix

```

```

14227: // TPackedRotationMatrix = array [0..2] of SmallInt;
14228: Function glTexPointMake( const s, t : Single ) : TTexPoint
14229: Function glAffineVectorMake( const x, y, z : Single ) : TAffineVector;
14230: Function glAffineVectorMake1( const v : TVectorGL ) : TAffineVector;
14231: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single );
14232: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single );
14233: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL );
14234: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector );
14235: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector );
14236: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL );
14237: Function glVectorMake( const v : TAffineVector; w : Single ) : TVectorGL;
14238: Function glVectorMake1( const x, y, z : Single; w : Single ) : TVectorGL;
14239: Function glPointMake( const x, y, z : Single ) : TVectorGL;
14240: Function glPointMake1( const v : TAffineVector ) : TVectorGL;
14241: Function glPointMake2( const v : TVectorGL ) : TVectorGL;
14242: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single );
14243: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single );
14244: Procedure glg1SetVector7( var v : TVectorGL; const vSrc : TVectorGL );
14245: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single );
14246: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector );
14247: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL );
14248: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single );
14249: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single );
14250: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector );
14251: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL );
14252: Procedure glRstVector( var v : TAffineVector );
14253: Procedure glRstVector1( var v : TVectorGL );
14254: Function glVectorAdd( const v1, v2 : TAffineVector ) : TAffineVector;
14255: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector );
14256: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector );
14257: Function glVectorAdd3( const v1, v2 : TVectorGL ) : TVectorGL;
14258: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL );
14259: Function glVectorAdd5( const v : TAffineVector; const f : Single ) : TAffineVector;
14260: Function glVectorAdd6( const v : TVectorGL; const f : Single ) : TVectorGL;
14261: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector );
14262: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL );
14263: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL );
14264: Procedure glAddVector10( var v : TAffineVector; const f : Single );
14265: Procedure glAddVector11( var v : TVectorGL; const f : Single );
14266: //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Int;dest:PTexPointArray);
14267: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Integer;const scale: TTExPoint; dest : PTExPointArray );
14268: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest: PAffineVectorArray );
14269: Function glVectorSubtract( const V1, V2 : TAffineVector ) : TAffineVector;
14270: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector );
14271: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL );
14272: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL );
14273: Function glVectorSubtract4( const V1, V2 : TVectorGL ) : TVectorGL;
14274: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL );
14275: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector );
14276: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single ) : TAffineVector;
14277: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single ) : TVectorGL;
14278: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector );
14279: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL );
14280: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single );
14281: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat );
14282: Function glTexPointCombine( const t1, t2 : TTExPoint; f1, f2 : Single ) : TTExPoint
14283: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single ) : TAffineVector;
14284: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single ) : TAffineVector;
14285: Procedure glVectorCombine34( const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector );
14286: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single );
14287: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single );
14288: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single ) : TVectorGL;
14289: Function glVectorCombine8( const V1 : TVectorGL; const V2 : TAffineVector; const F1,F2:Single ) : TVectorGL;
14290: Procedure glVectorCombine9( const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL );
14291: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL );
14292: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL );
14293: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single ) : TVectorGL;
14294: Procedure glVectorCombine31( const V1, V2, V3:TVectorGL; const F1,F2,F3:Single; var vr : TVectorGL );
14295: Function glVectorDotProduct( const V1, V2 : TAffineVector ) : Single;
14296: Function glVectorDotProduct1( const V1, V2 : TVectorGL ) : Single;
14297: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector ) : Single;
14298: Function glPointProject( const p, origin, direction : TAffineVector ) : Single;
14299: Function glPointProject1( const p, origin, direction : TVectorGL ) : Single;
14300: Function glVectorCrossProduct( const V1, V2 : TAffineVector ) : TAffineVector;
14301: Function glVectorCrossProduct1( const V1, V2 : TVectorGL ) : TVectorGL;
14302: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL );
14303: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL );
14304: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector );
14305: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector );
14306: Function glLerp( const start, stop, t : Single ) : Single
14307: Function glAngleLerp( start, stop, t : Single ) : Single
14308: Function glDistanceBetweenAngles( angle1, angle2 : Single ) : Single
14309: Function glTexPointLerp( const t1, t2 : TTExPoint; t : Single ) : TTExPoint;
14310: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14311: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector );
14312: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single ) : TVectorGL;

```

```

14313: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14314: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14315: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14316: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14317: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
PAffineVectorArray);
14318: Function glVectorLength( const x, y : Single) : Single;
14319: Function glVectorLength1( const x, y, z : Single) : Single;
14320: Function glVectorLength2( const v : TAffineVector) : Single;
14321: Function glVectorLength3( const v : TVectorGL) : Single;
14322: Function glVectorLength4( const v : array of Single) : Single;
14323: Function glVectorNorm( const x, y : Single) : Single;
14324: Function glVectorNorm1( const v : TAffineVector) : Single;
14325: Function glVectorNorm2( const v : TVectorGL) : Single;
14326: Function glVectorNorm3( var V : array of Single) : Single;
14327: Procedure glNormalizeVector( var v : TAffineVector);
14328: Procedure glNormalizeVector1( var v : TVectorGL);
14329: Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14330: Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14331: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14332: Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single;
14333: Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14334: Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14335: Procedure glNegateVector( var V : TAffineVector);
14336: Procedure glNegateVector2( var V : TVectorGL);
14337: Procedure glNegateVector3( var V : array of Single);
14338: Procedure glScaleVector( var v : TAffineVector; factor : Single);
14339: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14340: Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14341: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14342: Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14343: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14344: Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14345: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14346: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14347: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14348: Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14349: Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14350: Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14351: Function glVectorIsNull( const v : TVectorGL) : Boolean;
14352: Function glVectorIsNotNull( const v : TAffineVector) : Boolean;
14353: Function glVectorSpacing( const v1, v2 : TTExPoint) : Single;
14354: Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14355: Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14356: Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14357: Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14358: Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14359: Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14360: Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector;
14361: Function glVectorReflect( const V, N : TAffineVector) : TAffineVector;
14362: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14363: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14364: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single);
14365: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14366: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14367: Procedure glVectorRotateAroundY1( const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14368: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14369: Procedure glAbsVector( var v : TVectorGL);
14370: Procedure glAbsVector1( var v : TAffineVector);
14371: Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14372: Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14373: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14374: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14375: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14376: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14377: Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14378: Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14379: Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14380: Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14381: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14382: Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14383: Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14384: Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14385: Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14386: Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14387: Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14388: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14389: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14390: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix;
14391: Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14392: Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14393: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14394: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14395: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14396: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14397: Function glVectorTransform3(const V : TAffineVector; const M : TAffineMatrix):TAffineVector;
14398: Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14399: Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14400: Procedure glAdjointMatrix( var M : TMatrixGL);

```

```

14401: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14402: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14403: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14404: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14405: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14406: Procedure glNormalizeMatrix( var M : TMatrixGL)
14407: Procedure glTransposeMatrix( var M : TAffineMatrix);
14408: Procedure glTransposeMatrix1( var M : TMatrixGL);
14409: Procedure glInvertMatrix( var M : TMatrixGL);
14410: Procedure glInvertMatrix1( var M : TAffineMatrix);
14411: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14412: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14413: Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14414: Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14415: Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14416: Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14417: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14418: Procedure glNormalizePlane( var plane : THmgPlane)
14419: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14420: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14421: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14422: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14423: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14424: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14425: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14426: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14427: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14428: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14429: Function glPointSegmentDistance( const point, segmentStart, segmentStop:TAffineVector) : single
14430: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14431: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14432: Procedure SglementSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
Segment0Closest, Segment1Closest : TAffineVector)
14433: Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14434: TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14435: Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14436: Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14437: Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14438: Procedure glNormalizeQuaternion( var Q : TQuaternion)
14439: Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14440: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14441: Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14442: Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14443: Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14444: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14445: Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14446: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14447: Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14448: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14449: Function glQuaternionSlerp1( const source, dest:TQuaternion; const t: Single) : TQuaternion;
14450: Function glLnXP1( X : Extended) : Extended
14451: Function glLog10( X : Extended) : Extended
14452: Function glLog2( X : Extended) : Extended;
14453: Function glLog21( X : Single) : Single;
14454: Function glLogN( Base, X : Extended) : Extended
14455: Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14456: Function glPower( const Base, Exponent : Single) : Single;
14457: Function glPower1( Base : Single; Exponent : Integer) : Single;
14458: Function glDegToRad( const Degrees : Extended) : Extended;
14459: Function glDegToRad1( const Degrees : Single) : Single;
14460: Function glRadToDeg( const Radians : Extended) : Extended;
14461: Function glRadToDeg1( const Radians : Single) : Single;
14462: Function glNormalizeAngle( angle : Single) : Single
14463: Function glNormalizeDegAngle( angle : Single) : Single
14464: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14465: Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14466: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14467: Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14468: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14469: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14470: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14471: Function glArcCos( const X : Extended) : Extended;
14472: Function glArcCos1( const x : Single) : Single;
14473: Function glArcSin( const X : Extended) : Extended;
14474: Function glArcSin1( const X : Single) : Single;
14475: Function glArcTan21( const Y, X : Extended) : Extended;
14476: Function glArcTan21( const Y, X : Single) : Single;
14477: Function glFastArcTan2( y, x : Single) : Single
14478: Function glTan( const X : Extended) : Extended;
14479: Function glTan1( const X : Single) : Single;
14480: Function glCoTan( const X : Extended) : Extended;
14481: Function glCoTan1( const X : Single) : Single;
14482: Function glSinh( const x : Single) : Single;
14483: Function glSinh1( const x : Double) : Double;
14484: Function glCosh( const x : Single) : Single;
14485: Function glCosh1( const x : Double) : Double;
14486: Function glRSqrt( v : Single) : Single
14487: Function glRLength( x, y : Single) : Single
14488: Function glISqrt( i : Integer) : Integer

```

```

14489: Function glILength( x, y : Integer ) : Integer;
14490: Function glILength1( x, y, z : Integer ) : Integer;
14491: Procedure glRegisterBasedExp
14492: Procedure glRandomPointOnSphere( var p : TAffineVector)
14493: Function glRoundInt( v : Single ) : Single;
14494: Function glRoundInt1( v : Extended ) : Extended;
14495: Function glTrunc( v : Single ) : Integer;
14496: Function glTrunc64( v : Extended ) : Int64;
14497: Function glInt( v : Single ) : Single;
14498: Function glInt1( v : Extended ) : Extended;
14499: Function glFrac( v : Single ) : Single;
14500: Function glFrac1( v : Extended ) : Extended;
14501: Function glRound( v : Single ) : Integer;
14502: Function glRound64( v : Single ) : Int64;
14503: Function glRound641( v : Extended ) : Int64;
14504: Function glTrunc( X : Extended ) : Int64;
14505: Function glRound( X : Extended ) : Int64;
14506: Function glFrac( X : Extended ) : Extended;
14507: Function glCeil( v : Single ) : Integer;
14508: Function glCeil64( v : Extended ) : Int64;
14509: Function glFloor( v : Single ) : Integer;
14510: Function glFloor64( v : Extended ) : Int64;
14511: Function glScaleAndRound( i : Integer; var s : Single ) : Integer;
14512: Function glSign( x : Single ) : Integer;
14513: Function glIsInRange( const x, a, b : Single ) : Boolean;
14514: Function glIsInRange1( const x, a, b : Double ) : Boolean;
14515: Function glIsInCube( const p, d : TAffineVector ) : Boolean;
14516: Function glIsInCubel( const p, d : TVectorGL ) : Boolean;
14517: //Function MinFloat( values : PSingleArray; nbItems : Integer ) : Single;
14518: //Function MinFloat1( values : PDoubleArray; nbItems : Integer ) : Double;
14519: //Function MinFloat2( values : PExtendedArray; nbItems : Integer ) : Extended;
14520: Function glMinFloat3( const v1, v2 : Single ) : Single;
14521: Function glMinFloat4( const v : array of Single ) : Single;
14522: Function glMinFloat5( const v1, v2 : Double ) : Double;
14523: Function glMinFloat6( const v1, v2 : Extended ) : Extended;
14524: Function glMinFloat7( const v1, v2, v3 : Single ) : Single;
14525: Function glMinFloat8( const v1, v2, v3 : Double ) : Double;
14526: Function glMinFloat9( const v1, v2, v3 : Extended ) : Extended;
14527: //Function MaxFloat10( values : PSingleArray; nbItems : Integer ) : Single;
14528: //Function MaxFloat( values : PDoubleArray; nbItems : Integer ) : Double;
14529: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer ) : Extended;
14530: Function glMaxFloat2( const v : array of Single ) : Single;
14531: Function glMaxFloat3( const v1, v2 : Single ) : Single;
14532: Function glMaxFloat4( const v1, v2 : Double ) : Double;
14533: Function glMaxFloat5( const v1, v2 : Extended ) : Extended;
14534: Function glMaxFloat6( const v1, v2, v3 : Single ) : Single;
14535: Function glMaxFloat7( const v1, v2, v3 : Double ) : Double;
14536: Function glMaxFloat8( const v1, v2, v3 : Extended ) : Extended;
14537: Function glMinInteger9( const v1, v2 : Integer ) : Integer;
14538: Function glMinInteger( const v1, v2 : Cardinal ) : Cardinal;
14539: Function glMaxInteger( const v1, v2 : Integer ) : Integer;
14540: Function glMaxInteger1( const v1, v2 : Cardinal ) : Cardinal;
14541: Function glTriangleArea( const p1, p2, p3 : TAffineVector ) : Single;
14542: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14543: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector ) : Single;
14544: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14545: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single );
14546: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single );
14547: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single );
14548: Procedure gloffsetFloatArray1( var values : array of Single; delta : Single );
14549: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer );
14550: Function glMaxXYZComponent( const v : TVectorGL ) : Single;
14551: Function glMaxXYZComponent1( const v : TAffineVector ) : single;
14552: Function glMinXYZComponent( const v : TVectorGL ) : Single;
14553: Function glMinXYZComponent1( const v : TAffineVector ) : single;
14554: Function glMaxAbsXYZComponent( v : TVectorGL ) : Single;
14555: Function glMinAbsXYZComponent( v : TVectorGL ) : Single;
14556: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL );
14557: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector );
14558: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL );
14559: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector );
14560: Procedure glSortArrayAscending( var a : array of Extended );
14561: Function glClampValue( const aValue, aMin, aMax : Single ) : Single;
14562: Function glClampValue1( const aValue, aMin : Single ) : Single;
14563: Function glGeometryOptimizationMode : String;
14564: Procedure glBeginFPUOnlySection;
14565: Procedure glEndFPUOnlySection;
14566: Function glConvertRotation( const Angles : TAffineVector ) : TVectorGL;
14567: Function glMakeAffineDblVector( var v : array of Double ) : TAffineDblVector;
14568: Function glMakeDblVector( var v : array of Double ) : THomogeneousDblVector;
14569: Function glVectorOrffineDblToFlt( const v : TAffineDblVector ) : TAffineVector;
14570: Function glVectorDblToFlt( const v : THomogeneousDblVector ) : THomogeneousVector;
14571: Function glVectorOrffineFltToDbl( const v : TAffineVector ) : TAffineDblVector;
14572: Function glVectorFltToDbl( const v : TVectorGL ) : THomogeneousDblVector;
14573: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single ) : Boolean;
14574: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word );
14575: Function glTurn( const Matrix : TMatrixGL; angle : Single ) : TMatrixGL;
14576: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single ):TMatrixGL;
14577: Function glPitch( const Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;

```

```

14578: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14579: Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14580: Function glRoll1(const Matrix:TMatrixGL;const MasterDirection:TAffineVector;Angle:Single): TMatrixGL;
14581: Function glRayCastMinDistToPoint(const rayStart,rayVector:TVectorGL;const point:TVectorGL):Single
14582: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
sphereCenter:TVectorGL;const sphereRadius : Single ) : Boolean;
14583: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
const sphereRadius : Single; var i1, i2 : TVectorGL ) : Integer;
14584: Function glSphereVisibleRadius( distance, radius : Single ) : Single
14585: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL ) : TFrustum
14586: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14587: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14588: Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci: TRenderContextClippingInfo):Bool;
14589: Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
Frustum:TFrustum):Bool;
14590: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL ) : TMatrixGL
14591: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL ) : TMatrixGL
14592: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector ) : TMatrixGL
14593: Function glPackRotationMatrix( const mat : TMatrixGL ) : TPackedRotationMatrix
14594: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix ) : TMatrixGL
14595: 'cPI','Single').setExtended( 3.141592654 );
14596: 'cPIdiv180','Single').setExtended( 0.017453292 );
14597: 'c180divPI','Single').setExtended( 57.29577951 );
14598: 'c2PI','Single').setExtended( 6.283185307 );
14599: 'cPIdiv2','Single').setExtended( 1.570796326 );
14600: 'cPIdiv4','Single').setExtended( 0.785398163 );
14601: 'c3PIdiv4','Single').setExtended( 2.35619449 );
14602: 'cInv2PI','Single').setExtended( 1 / 6.283185307 );
14603: 'cInv360','Single').setExtended( 1 / 360 );
14604: 'c180','Single').setExtended( 180 );
14605: 'c360','Single').setExtended( 360 );
14606: 'cOneHalf','Single').setExtended( 0.5 );
14607: 'cLn10','Single').setExtended( 2.302585093 );
14608: ('MinSingle','Extended').setExtended( 1.5e-45 );
14609: 'MaxSingle','Extended').setExtended( 3.4e+38 );
14610: 'MinDouble','Extended').setExtended( 5.0e-324 );
14611: 'MaxDouble','Extended').setExtended( 1.7e+308 );
14612: 'MinExtended','Extended').setExtended( 3.4e-4932 );
14613: 'MaxExtended','Extended').setExtended( 1.1e+4932 );
14614: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18 );
14615: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18 );
14616: end;
14617:
14618: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14619: begin
14620: AddClassN(FindClass('TOBJECT'), 'TMeshObjectList'
14621: (FindClass('TOBJECT'))'TFaceGroups
14622: TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14623: TMeshAutoCenterings', 'set of TMeshAutoCentering
14624: TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14625: SIRegister_TBaseMeshObject(CL);
14626: (FindClass('TOBJECT'))'TSkeletonFrameList
14627: TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14628: SIRegister_TSkeletonFrame(CL);
14629: SIRegister_TSkeletonFrameList(CL);
14630: (FindClass('TOBJECT'))'TSkeleton
14631: (FindClass('TOBJECT'))'TSkeletonBone
14632: SIRegister_TSkeletonBoneList(CL);
14633: SIRegister_TSkeletonRootBoneList(CL);
14634: SIRegister_TSkeletonBone(CL);
14635: (FindClass('TOBJECT'))'TSkeletonColliderList
14636: SIRegister_TSkeletonCollider(CL);
14637: SIRegister_TSkeletonColliderList(CL);
14638: (FindClass('TOBJECT'))'TGLBaseMesh
14639: TBlendLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14640: +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14641: +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14642: +'QuaternionList; end
14643: SIRegister_TSkeleton(CL);
14644: TMeshObjectRenderingOption', '( moroGroupByMaterial )
14645: TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14646: SIRegister_TMeshObject(CL);
14647: SIRegister_TMeshObjectList(CL);
14648: //TMeshObjectListClass', 'class of TMeshObjectList
14649: (FindClass('TOBJECT'))'TMeshMorphTargetList
14650: SIRegister_TMorphTarget(CL);
14651: SIRegister_TMorphTargetList(CL);
14652: SIRegister_TMorphableMeshObject(CL);
14653: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14654: //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14655: //VerticesBoneWeights', '^TVerticesBoneWeights // will not work
14656: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14657: SIRegister_TSkeletonMeshObject(CL);
14658: SIRegister_TFaceGroup(CL);
14659: TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14660: +'atTriangles, fgmmTriangleFan, fgmmQuads )
14661: SIRegister_TFGVertexIndexList(CL);

```

```

14662: SIRегистер_TFGVertexNormalTexIndexList(CL);
14663: SIRегистер_TFGIndexTexCoordList(CL);
14664: SIRегистер_TFaceGroups(CL);
14665: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14666: SIRегистер_TVectorFile(CL);
14667: //TVectorFileClass', 'class of TVectorFile
14668: SIRегистер_TGLGLSMVectorFile(CL);
14669: SIRегистер_TGLBaseMesh(CL);
14670: SIRегистер_TGLFreeForm(CL);
14671: TGLActorOption', '( aoSkeletonNormalizeNormals )
14672: TGLActorOptions', 'set of TGLActorOption
14673: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14674: (FindClass('TOBJECT'), 'TGLActor
14675: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14676: SIRегистер_TActorAnimation(CL);
14677: TActorAnimationName', 'String
14678: SIRегистер_TActorAnimations(CL);
14679: SIRегистер_TGLBaseAnimationController(CL);
14680: SIRегистер_TGLAnimationController(CL);
14681: TActorFrameInterpolation', '( afpNone, afpLinear )
14682: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14683: +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14684: SIRегистер_TGLActor(CL);
14685: SIRегистер_TVectorFileFormat(CL);
14686: SIRегистер_TVectorFileFormatsList(CL);
14687: (FindClass('TOBJECT'), 'EInvalidVectorFile
14688: Function GetVectorFileFormats : TVectorFileFormatsList
14689: Function VectorFileFormatsFilter : String
14690: Function VectorFileFormatsSaveFilter : String
14691: Function VectorFileFormatExtensionByIndex( index : Integer ) : String
14692: Procedure RegisterVectorFileFormat( const aExtension, aDescription: String; aClass : TVectorFileClass)
14693: Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14694: end;
14695:
14696: procedure SIRегистер_AxCtrls(CL: TPSPascalCompiler);
14697: begin
14698: 'Class_DColorPropPage', 'GUID {5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14699: 'Class_DFontPropPage', 'GUID {5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14700: 'Class_DPicturePropPage', 'GUID {5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14701: 'Class_DStringPropPage', 'GUID {F42D677E-754B-11D0-BDFB-00A024D1875C}
14702: SIRегистер_TOLEStream(CL);
14703: (FindClass('TOBJECT'), 'TConnectionPoints
14704: TConnectionKind', '( ckSingle, ckMulti )
14705: SIRегистер_TConnectionPoint(CL);
14706: SIRегистер_TConnectionPoints(CL);
14707: TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14708: (FindClass('TOBJECT'), 'TActiveXControlFactory
14709: SIRегистер_TActiveXControl(CL);
14710: //TActiveXControlClass', 'class of TActiveXControl
14711: SIRегистер_TActiveXControlFactory(CL);
14712: SIRегистер_TActiveFormControl(CL);
14713: SIRегистер_TActiveForm(CL);
14714: //TActiveFormClass', 'class of TActiveForm
14715: SIRегистер_TActiveFormFactory(CL);
14716: (FindClass('TOBJECT'), 'TPropertyPageImpl
14717: SIRегистер_TPropertyPage(CL);
14718: //TPropertyPageClass', 'class of TPropertyPage
14719: SIRегистер_TPropertyPageImpl(CL);
14720: SIRегистер_TActiveXPropertyPage(CL);
14721: SIRегистер_TActiveXPropertyPageFactory(CL);
14722: SIRегистер_TCustomAdapter(CL);
14723: SIRегистер_TAdapterNotifier(CL);
14724: SIRегистер_IFontAccess(CL);
14725: SIRегистер_TFontAdapter(CL);
14726: SIRегистер_IPictureAccess(CL);
14727: SIRегистер_TPictureAdapter(CL);
14728: SIRегистер_TOLEGraphic(CL);
14729: SIRегистер_TStringsAdapter(CL);
14730: SIRегистер_TReflectorWindow(CL);
14731: Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:GUID;VTCode:Int;PropList:TStrings);
14732: Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14733: Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14734: Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14735: Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14736: Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14737: Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14738: Function ParkingWindow : HWND
14739: end;
14740:
14741: procedure SIRегистер_synaip(CL: TPSPascalCompiler);
14742: begin
14743: // TIP6Bytes = array [0..15] of Byte;
14744: (:binary form of IPv6 adress (for string conversion routines))
14745: // TIP6Words = array [0..7] of Word;
14746: AddTypeS('TIP6Bytes', 'array [0..15] of Byte;');
14747: AddTypeS('TIP6Words', 'array [0..7] of Word;');
14748: AddTypeS('TIPAddr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14749: Function synaIsIP( const Value : string) : Boolean';
14750: Function synaIsIP6( const Value : string) : Boolean';

```

```

14751: Function synaIPToID( Host : string ) : AnsiString';
14752: Function synaStrToIp6( value : string ) : TIp6Bytes';
14753: Function synaIp6ToStr( value : TIp6Bytes ) : string';
14754: Function synaStrToIp( value : string ) : integer';
14755: Function synaIpToStr( value : integer ) : string');
14756: Function synaReverseIP( Value : AnsiString ) : AnsiString';
14757: Function synaReverseIP6( Value : AnsiString ) : AnsiString');
14758: Function synaExpandIP6( Value : AnsiString ) : AnsiString');
14759: Function xStrToIP( const Value : String ) : TIPAdr');
14760: Function xIPToStr( const Adresse : TIPAdr ) : String');
14761: Function IPToCardinal( const Adresse : TIPAdr ) : Cardinal');
14762: Function CardinalToIP( const Value : Cardinal ) : TIPAdr');
14763: end;
14764:
14765: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14766: begin
14767: AddTypeS('TSpecials', 'set of Char');
14768: Const ('SpecialChar','TSpecials').SetSet( '='[]<>;@/?\"_');
14769: Const ('URLFullSpecialChar','TSpecials').SetSet( '/?:@=&#+' );
14770: Const ('TableBase64'ABCDEFGHJKLMNOPQRSTUVWXYZabcdeghijklmnopqrstuvwxyz0123456789+/=');
14771: Const ('TableBase64mod'ABCDEFGHJKLMNOPQRSTUVWXYZabcdeghijklmnopqrstuvwxyz0123456789+,=');
14772: Const ('TableUU'(!#$%&!)*+,-./0123456789;:<>?ABCDEFGHJKLMNOPQRSTUVWXYZ[\]^_');
14773: Const ('TableXX'(+0123456789ABCDEFGHJKLMNOPQRSTUVWXYZabcdeghijklmnopqrstuvwxyz^);
14774: Function DecodeTriplet( const Value : AnsiString; Delimiter : Char ) : AnsiString');
14775: Function DecodeQuotedPrintable( const Value : AnsiString ) : AnsiString');
14776: Function DecodeURL( const Value : AnsiString ) : AnsiString');
14777: Function EncodeTriplet(const Value:AnsiString;Delimiter:Char;Specials:TSpecials):AnsiString);
14778: Function EncodeQuotedPrintable( const Value : AnsiString ) : AnsiString');
14779: Function EncodeSafeQuotedPrintable( const Value : AnsiString ) : AnsiString');
14780: Function EncodeURLElement( const Value : AnsiString ) : AnsiString');
14781: Function EncodeURL( const Value : AnsiString ) : AnsiString');
14782: Function Decode4to3( const Value, Table : AnsiString ) : AnsiString');
14783: Function Decode4to3Ex( const Value, Table : AnsiString ) : AnsiString');
14784: Function Encode3to4( const Value, Table : AnsiString ) : AnsiString');
14785: Function synDecodeBase64( const Value : AnsiString ) : AnsiString');
14786: Function synEncodeBase64( const Value : AnsiString ) : AnsiString');
14787: Function DecodeBase64mod( const Value : AnsiString ) : AnsiString');
14788: Function EncodeBase64mod( const Value : AnsiString ) : AnsiString');
14789: Function DecodeUU( const Value : AnsiString ) : AnsiString');
14790: Function EncodeUU( const Value : AnsiString ) : AnsiString');
14791: Function DecodeXXX( const Value : AnsiString ) : AnsiString');
14792: Function DecodeYEnc( const Value : AnsiString ) : AnsiString');
14793: Function UpdateCrc32( Value : Byte; Crc32 : Integer ) : Integer');
14794: Function synCrc32( const Value : AnsiString ) : Integer');
14795: Function UpdateCrc16( Value : Byte; Crc16 : Word ) : Word');
14796: Function Crc16( const Value : AnsiString ) : Word');
14797: Function synMD5( const Value : AnsiString ) : AnsiString');
14798: Function HMAC_MD5( Text, Key : AnsiString ) : AnsiString');
14799: Function MD5LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14800: Function synSHA1( const Value : AnsiString ) : AnsiString');
14801: Function HMAC_SHA1( Text, Key : AnsiString ) : AnsiString');
14802: Function SHA1LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14803: Function synMD4( const Value : AnsiString ) : AnsiString');
14804: end;
14805:
14806: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14807: begin
14808: AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14809: +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14810: +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14811: +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14812: +'8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE, '
14813: +'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14814: +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14815: +', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14816: +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14817: +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14818: +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14819: +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14820: +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14821: +' , CP864, CP865, CP869, CP1125 ') );
14822: AddTypeS('TMimeSetChar', 'set of TMimeChar');
14823: Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14824: Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
  TransformTable : array of Word) : AnsiString');
14825: Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
  TransformTable : array of Word; Translit : Boolean) : AnsiString');
14826: Function GetCurCP : TMimeChar');
14827: Function GetCuroEMCP : TMimeChar');
14828: Function GetCPFromID( Value : AnsiString ) : TMimeChar');
14829: Function GetIDFromCP( Value : TMimeChar ) : AnsiString');
14830: Function NeedCharsetConversion( const Value : AnsiString ) : Boolean');
14831: Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14832: Function GetBOM( Value : TMimeChar ) : AnsiString');
14833: Function StringToWide( const Value : AnsiString ) : WideString');
14834: Function WideToString( const Value : WideString ) : AnsiString');
14835: end;
14836:
14837: procedure SIRegister_synamisc(CL: TPSPascalCompiler);

```

```

14838: begin
14839:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14840:   Procedure WakeOnLan( MAC, IP : string)';
14841:   Function GetDNS : string');
14842:   Function GetIEProxy( protocol : string) : TProxySetting');
14843:   Function GetLocalIPs : string');
14844: end;
14845:
14846:
14847: procedure SIRegister_synaser(CL: TPPSPascalCompiler);
14848: begin
14849:   AddConstantN('synCR', 'Char #$0d);
14850:   Const ('synLF', 'Char #$0a);
14851:   Const ('cSerialChunk', 'LongInt'( 8192);
14852:   Const ('LockfileDirectory', 'String '/var/lock');
14853:   Const ('PortIsClosed', 'LongInt'( - 1);
14854:   Const ('ErrAlreadyOwned', 'LongInt'( 9991);
14855:   Const ('ErrAlreadyInUse', 'LongInt'( 9992);
14856:   Const ('ErrWrongParameter', 'LongInt'( 9993);
14857:   Const ('ErrPortNotOpen', 'LongInt'( 9994);
14858:   Const ('ErrNoDeviceAnswer', 'LongInt'( 9995);
14859:   Const ('ErrMaxBuffer', 'LongInt'( 9996);
14860:   Const ('ErrTimeout', 'LongInt'( 9997);
14861:   Const ('ErrNotRead', 'LongInt'( 9998);
14862:   Const ('ErrFrame', 'LongInt'( 9999);
14863:   Const ('ErrOverrun', 'LongInt'( 10000);
14864:   Const ('ErrRxOver', 'LongInt'( 10001);
14865:   Const ('ErrRxParity', 'LongInt'( 10002);
14866:   Const ('ErrTxFull', 'LongInt'( 10003);
14867:   Const ('dcb_Binary', 'LongWord')( $00000001);
14868:   Const ('dcb_ParityCheck', 'LongWord')( $00000002);
14869:   Const ('dcb_OutxCtsFlow', 'LongWord')( $00000004);
14870:   Const ('dcb_OutxDsrFlow', 'LongWord')( $00000008);
14871:   Const ('dcb_DtrControlMask', 'LongWord')( $00000030);
14872:   Const ('dcb_DtrControlDisable', 'LongWord')( $00000000);
14873:   Const ('dcb_DtrControlEnable', 'LongWord')( $00000010);
14874:   Const ('dcb_DtrControlHandshake', 'LongWord')( $00000020);
14875:   Const ('dcb_DsrSensitivity', 'LongWord')( $00000040);
14876:   Const ('dcb_TXContinueOnXoff', 'LongWord')( $00000080);
14877:   Const ('dcb_OutX', 'LongWord')( $00000100);
14878:   Const ('dcb_InX', 'LongWord')( $00000200);
14879:   Const ('dcb_ErrorChar', 'LongWord')( $00000400);
14880:   Const ('dcb_NullStrip', 'LongWord')( $00000800);
14881:   Const ('dcb_RtsControlMask', 'LongWord')( $00003000);
14882:   Const ('dcb_RtsControlDisable', 'LongWord')( $00000000);
14883:   Const ('dcb_RtsControlEnable', 'LongWord')( $00001000);
14884:   Const ('dcb_RtsControlHandshake', 'LongWord')( $00002000);
14885:   Const ('dcb_RtsControlToggle', 'LongWord')( $00003000);
14886:   Const ('dcb_AbortOnError', 'LongWord')( $00004000);
14887:   Const ('dcb_Reserves', 'LongWord')( $FFFF8000);
14888:   Const ('synSB1', 'LongInt'( 0);
14889:   Const ('SB1andHalf', 'LongInt'( 1);
14890:   Const ('synSB2', 'LongInt'( 2);
14891:   Const ('synINVALID_HANDLE_VALUE', 'LongInt'( THandle ( - 1 ));
14892:   Const ('CS7fix', 'LongWord')( $0000020);
14893:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14894:     +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14895:     +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14896:     +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14897:   //AddTypeS('PDCB', '^TDCB // will not work');
14898:   //Const('MaxRates', 'LongInt'( 18);
14899:   //Const('MaxRates', 'LongInt'( 30);
14900:   //Const('MaxRates', 'LongInt'( 19);
14901:   Const ('_SYNC', 'LongWord')( $0080);
14902:   Const ('synOK', 'LongInt'( 0);
14903:   Const ('synErr', 'LongInt'( integer ( - 1 ));
14904:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
14905:   HR_WriteCount, HR_Wait )');
14906:   Type ('THookSerialStatus', Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14907:   SIRegister_ESynaserError(CL);
14908:   SIRegister_TBlockSerial(CL);
14909:   Function GetSerialPortNames : string);
14910: end;
14911: procedure SIRegister_synaicnv(CL: TPPSPascalCompiler);
14912: begin
14913:   Const ('DLLIconvName', 'String 'libiconv.so');
14914:   Const ('DLLIconvName', 'String 'iconv.dll');
14915:   AddTypeS('size_t', 'Cardinal');
14916:   AddTypeS('iconv_t', 'Integer');
14917:   //AddTypeS('iconv_t', 'Pointer');
14918:   AddTypeS('argptr', 'iconv_t');
14919:   Function SynalconvOpen( const tocode, fromcode : Ansistring) : iconv_t';
14920:   Function SynalconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t';
14921:   Function SynalconvOpenIgnore( const tocode, fromcode : AnsiString) : iconv_t';
14922:   Function Synalconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer';
14923:   Function SynalconvClose( var cd : iconv_t) : integer';
14924:   Function SynalconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer';
14925:   Function IsIconvloaded : Boolean';

```

```

14926: Function InitIconvInterface : Boolean');
14927: Function DestroyIconvInterface : Boolean');
14928: Const ('ICONV_TRIVIALP','LongInt'( 0);
14929: Const ('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14930: Const ('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14931: Const ('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14932: Const ('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14933: end;
14934:
14935: procedure SIRegister_pingsend(CL: TPPSPascalCompiler);
14936: begin
14937: Const 'ICMP_ECHO','LongInt'( 8);
14938: Const ('ICMP_ECHOREPLY','LongInt'( 0);
14939: Const ('ICMP_UNREACH','LongInt'( 3);
14940: Const ('ICMP_TIME_EXCEEDED','LongInt'( 11);
14941: Const ('ICMP6_ECHO','LongInt'( 128);
14942: Const ('ICMP6_ECHOREPLY','LongInt'( 129);
14943: Const ('ICMP6_UNREACH','LongInt'( 1);
14944: Const ('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14945: AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt' +
14946: +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14947: SIRegister_TPINGSend(CL);
14948: Function PingHost( const Host : string) : Integer';
14949: Function TraceRouteHost( const Host : string) : string');
14950: end;
14951:
14952: procedure SIRegister_asn1util(CL: TPPSPascalCompiler);
14953: begin
14954: AddConstantN('synASN1_BOOL','LongWord')($01);
14955: Const ('synASN1_INT','LongWord')($02);
14956: Const ('synASN1_OCTSTR','LongWord')($04);
14957: Const ('synASN1_NULL','LongWord')($05);
14958: Const ('synASN1_OBJID','LongWord')($06);
14959: Const ('synASN1_ENUM','LongWord')($0a);
14960: Const ('synASN1_SEQ','LongWord')($30);
14961: Const ('synASN1_SETOF','LongWord')($31);
14962: Const ('synASN1_IPADDR','LongWord')($40);
14963: Const ('synASN1_COUNTER','LongWord')($41);
14964: Const ('synASN1_GAUGE','LongWord')($42);
14965: Const ('synASN1_TIMETICKS','LongWord')($43);
14966: Const ('synASN1_OPAQUE','LongWord')($44);
14967: Function synASNEncOIDItem( Value : Integer) : AnsiString';
14968: Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer';
14969: Function synASNEncLen( Len : Integer) : AnsiString';
14970: Function synASNEncLen( var Start : Integer; const Buffer : AnsiString) : Integer';
14971: Function synASNEncInt( Value : Integer) : AnsiString';
14972: Function synASNEncCUInt( Value : Integer) : AnsiString';
14973: Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString';
14974: Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14975: Function synMibToId( Mib : String) : AnsiString';
14976: Function synIdToMib( const Id : AnsiString) : String';
14977: Function synIntMibToStr( const Value : AnsiString) : AnsiString';
14978: Function ASNdump( const Value : AnsiString) : AnsiString';
14979: Function GetMailServers(const DNSHost, Domain : AnsiString; const Servers:TStrings):Bool';
14980: Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString';
14981: end;
14982:
14983: procedure SIRegister_ldapsend(CL: TPPSPascalCompiler);
14984: begin
14985: Const ('cLDAPProtocol','String '389');
14986: Const ('LDAP ASN1 BIND REQUEST','LongWord')($60);
14987: Const ('LDAP ASN1 BIND RESPONSE','LongWord')($61);
14988: Const ('LDAP ASN1 UNBIND REQUEST','LongWord')($42);
14989: Const ('LDAP ASN1 SEARCH REQUEST','LongWord')($63);
14990: Const ('LDAP ASN1 SEARCH ENTRY','LongWord')($64);
14991: Const ('LDAP ASN1 SEARCH DONE','LongWord')($65);
14992: Const ('LDAP ASN1 SEARCH REFERENCE','LongWord')($73);
14993: Const ('LDAP ASN1 MODIFY REQUEST','LongWord')($66);
14994: Const ('LDAP ASN1 MODIFY RESPONSE','LongWord')($67);
14995: Const ('LDAP ASN1 ADD REQUEST','LongWord')($68);
14996: Const ('LDAP ASN1 ADD RESPONSE','LongWord')($69);
14997: Const ('LDAP ASN1 DEL REQUEST','LongWord')($4A);
14998: Const ('LDAP ASN1 DEL RESPONSE','LongWord')($6B);
14999: Const ('LDAP ASN1 MODIFYDN REQUEST','LongWord')($6C);
15000: Const ('LDAP ASN1 MODIFYDN RESPONSE','LongWord')($6D);
15001: Const ('LDAP ASN1 COMPARE REQUEST','LongWord')($6E);
15002: Const ('LDAP ASN1 COMPARE RESPONSE','LongWord')($6F);
15003: Const ('LDAP ASN1 ABANDON REQUEST','LongWord')($70);
15004: Const ('LDAP ASN1 EXT REQUEST','LongWord')($77);
15005: Const ('LDAP ASN1 EXT RESPONSE','LongWord')($78);
15006: SIRegister_TLDAPAttribute(CL);
15007: SIRegister_TLDAPAttributeList(CL);
15008: SIRegister_TLDAPResult(CL);
15009: SIRegister_TLDAPResultList(CL);
15010: AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
15011: AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
15012: AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
15013: SIRegister_TLDAPSnd(CL);
15014: Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString';

```

```

15015: end;
15016:
15017:
15018: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
15019: begin
15020:   Const('cSysLogProtocol','String '514');
15021:   Const('FCL_Kernel','LongInt'( 0));
15022:   Const('FCL_UserLevel','LongInt'( 1));
15023:   Const('FCL_MailSystem','LongInt'( 2));
15024:   Const('FCL_System','LongInt'( 3));
15025:   Const('FCL_Security','LongInt'( 4));
15026:   Const('FCL_Syslogd','LongInt'( 5));
15027:   Const('FCL_Printer','LongInt'( 6));
15028:   Const('FCL_News','LongInt'( 7));
15029:   Const('FCL_UUCP','LongInt'( 8));
15030:   Const('FCL_Clock','LongInt'( 9));
15031:   Const('FCL_Authorization','LongInt'( 10));
15032:   Const('FCL_FTP','LongInt'( 11));
15033:   Const('FCL_NTP','LongInt'( 12));
15034:   Const('FCL_LogAudit','LongInt'( 13));
15035:   Const('FCL_LogAlert','LongInt'( 14));
15036:   Const('FCL_Time','LongInt'( 15));
15037:   Const('FCL_Local0','LongInt'( 16));
15038:   Const('FCL_Local1','LongInt'( 17));
15039:   Const('FCL_Local2','LongInt'( 18));
15040:   Const('FCL_Local3','LongInt'( 19));
15041:   Const('FCL_Local4','LongInt'( 20));
15042:   Const('FCL_Local5','LongInt'( 21));
15043:   Const('FCL_Local6','LongInt'( 22));
15044:   Const('FCL_Local7','LongInt'( 23));
15045:   Type(TSyslogSeverity)'(Emergency, Alert, Critical, Error, Warning, Notice, Info, Debug);
15046:   SIRegister_TSyslogMessage(CL);
15047:   SIRegister_TSyslogSend(CL);
15048:   Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string):Boolean;
15049: end;
15050:
15051:
15052: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
15053: begin
15054:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15055:   SIRegister_TMessHeader(CL);
15056:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
15057:   SIRegister_TMimeMess(CL);
15058: end;
15059:
15060: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
15061: begin
15062:   (FindClass('TOBJECT'),'TMimePart');
15063:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15064:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15065:   AddTypeS('TMimeEncoding', '(ME_7BIT,ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15066:   SIRegister_TMimePart(CL);
15067:   Const('MaxMimeType','LongInt'( 25));
15068:   Function GenerateBoundary : string';
15069: end;
15070:
15071: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15072: begin
15073:   Function InlineDecode( const Value : string; CP : TMimeChar) : string';
15074:   Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string';
15075:   Function NeedInline( const Value : AnsiString) : boolean';
15076:   Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string';
15077:   Function InlineCode( const Value : string) : string';
15078:   Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string';
15079:   Function InlineEmail( const Value : string) : string';
15080: end;
15081:
15082: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15083: begin
15084:   Const('cFtpProtocol','String '21');
15085:   Const('cFtpDataProtocol','String '20');
15086:   Const('FTP_OK','LongInt'( 255));
15087:   Const('FTP_ERR','LongInt'( 254));
15088:   AddTypeS('TFTPSStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15089:   SIRegister_TFTPListRec(CL);
15090:   SIRegister_TFTPList(CL);
15091:   SIRegister_TFTPSend(CL);
15092:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15093:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean';
15094:   Function FtpInterServerTransfer(const FromIP,FromPort,FromFile,FromUser,FromPass : string; const ToIP,
ToPort,ToFile,ToUser,ToPass : string) : Boolean';
15095: end;
15096:
15097: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15098: begin
15099:   Const('cHttpProtocol','String '80');
15100:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15101:   SIRegister_THTTPSend(CL);

```

```

15102: Function HttpGetText( const URL : string; const Response : TStrings ) : Boolean';
15103: Function HttpGetBinary( const URL : string; const Response : TStream ) : Boolean';
15104: Function HttpPostBinary( const URL : string; const Data : TStream ) : Boolean';
15105: Function HttpPostURL( const URL, URLData : string; const Data : TStream ) : Boolean';
15106: Function HttpPostFile( const URL,FieldName,FileName:string;const Data:TStream;const
  ResultData:TStrings ) :Bool;
15107: end;
15108:
15109: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15110: begin
15111:   Const('cSmtpProtocol','String '25');
15112:   SIRegister_TSMTPSend(CL);
15113:   Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
  Passw:string):Bool;
15114:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Bool;
15115:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
  Username, Password : string):Boolean';
15116: end;
15117:
15118: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15119: begin
15120:   Const('cSnmpProtocol','String '161');
15121:   Const('cSnmpTrapProtocol','String '162');
15122:   Const('SNMP_V1','LongInt'( 0 );
15123:   Const('SNMP_V2C','LongInt'( 1 );
15124:   Const('SNMP_V3','LongInt'( 3 );
15125:   Const('PDUGetRequest','LongWord')( $A0 );
15126:   Const('PDUGetNextRequest','LongWord')( $A1 );
15127:   Const('PDUGetResponse','LongWord')( $A2 );
15128:   Const('PDUSetRequest','LongWord')( $A3 );
15129:   Const('PDUTrap','LongWord')( $A4 );
15130:   Const('PDUGetBulkRequest','LongWord')( $A5 );
15131:   Const('PDUInformRequest','LongWord')( $A6 );
15132:   Const('PDUTrapV2','LongWord')( $A7 );
15133:   Const('PDUReport','LongWord')( $A8 );
15134:   Const('ENoError','LongInt 0;
15135:   Const('ETooBig','LongInt')( 1 );
15136:   Const('ENoSuchName','LongInt')( 2 );
15137:   Const('EBadValue','LongInt')( 3 );
15138:   Const('EReadOnly','LongInt')( 4 );
15139:   Const('EGenErr','LongInt')( 5 );
15140:   Const('ENoAccess','LongInt')( 6 );
15141:   Const('EWrongType','LongInt')( 7 );
15142:   Const('EWrongLength','LongInt')( 8 );
15143:   Const('EWrongStringEncoding','Longint')( 9 );
15144:   Const('EWrongValue','LongInt')( 10 );
15145:   Const('ENoCreation','LongInt')( 11 );
15146:   Const('EInconsistentValue','LongInt')( 12 );
15147:   Const('EResourceUnavailable','LongInt')( 13 );
15148:   Const('ECommitFailed','LongInt')( 14 );
15149:   Const('EUndoFailed','LongInt')( 15 );
15150:   Const('EAuthorizationError','LongInt')( 16 );
15151:   Const('ENotWritable','LongInt')( 17 );
15152:   Const('EInconsistentName','LongInt')( 18 );
15153:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15154:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15155:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15156:   SIRegister_TSNSMPMib(CL);
15157:   AddTypes('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
  + 'EngineTime : integer; EngineStamp : Cardinal; end');
15158:   SIRegister_TSNSMPRec(CL);
15159:   SIRegister_TSNSMPSend(CL);
15160:   SIRegister_TSNSMPSet(CL);
15161:   Function SNMPGet(const OID,Community,SNMPHost:AnsiString; var Value: AnsiString) : Boolean';
15162:   Function SNMPSet(const OID,Community,SNMPHost,Value:AnsiString;ValueType:Integer): Boolean';
15163:   Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15164:   Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15165:   Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:Ansistr;var Value:Ansistr):Bool;
15166:   Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds : Integer;
  const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer';
15167:   Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds : Integer;
  const MIBName, MIBValue : TStringList) : Integer';
15168: end;
15169:
15170: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15171: begin
15172:   Function GetDomainName2: AnsiString';
15173:   Function GetDomainController( Domain : AnsiString ) : AnsiString';
15174:   Function GetDomainUsers( Controller : AnsiString ) : AnsiString';
15175:   Function GetDomainGroups( Controller : AnsiString ) : AnsiString';
15176:   Function GetDateTime( Controller : AnsiString ) : TDateTime';
15177:   Function GetFullName2( Controller, UserID : AnsiString ) : AnsiString';
15178: end;
15179:
15180: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15181: begin
15182:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15183:   TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM)';
15184:   Function wwStrToDate( const S : string ) : boolean';
15185:   Function wwStrToTime( const S : string ) : boolean';

```

```

15186: Function wwStrToDateTime( const S : string ) : boolean';
15187: Function wwStrToTimeVal( const S : string ) : TDateTime';
15188: Function wwStrToDateVal( const S : string ) : TDateTime';
15189: Function wwStrToDateTimeVal( const S : string ) : TDateTime';
15190: Function wwStrToInt( const S : string ) : boolean';
15191: Function wwStrToFloat( const S : string ) : boolean';
15192: Function wwGetDateOrder( const DateFormat : string ) : TwwDateOrder';
15193: Function wwNextDay( Year, Month, Day : Word ) : integer';
15194: Function wwPriorDay( Year, Month, Day : Word ) : integer';
15195: Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime ) : Boolean';
15196: Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime ) : Boolean';
15197: Function wwGetDateTimeCursorPosition( SelStart:int;Text:string;TimeOnly:Bool ):TwwDateTimeSelection';
15198: Function wwGetTimeCursorPosition( SelStart : integer; Text : string ) : TwwDateTimeSelection';
15199: Function wwScanDate( const S : string; var Date : TDateTime ) : Boolean';
15200: Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer ) : Boolean';
15201: Procedure wwSetDateTimeCursorPosition(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15202: Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat : string ) : boolean';
15203: end;
15204:
15205: unit uPSI_Themes;
15206: Function ThemeServices : TThemeServices';
15207: Function ThemeControl( AControl : TControl ) : Boolean';
15208: Function UnthemedDesigner( AControl : TControl ) : Boolean';
15209: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15210: begin
15211:   Function GetBindingkeyAccessPoint( const Operator : String; const key : String ) : String';
15212: end;
15213: Unit uPSC_menus;
15214: Function StripHotkey( const Text : string ) : string';
15215: Function GetHotkey( const Text : string ) : string';
15216: Function AnsiSameCaption( const Text1, Text2 : string ) : Boolean';
15217: Function IsAltGRPressed : boolean';
15218:
15219: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15220: begin
15221:   TCommandEvent', 'Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15222:   SIRegister_TIdIMAP4Server(CL);
15223: end;
15224:
15225: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15226: begin
15227:   'HASH_SIZE','LongInt'( 256 );
15228:   CL.FindClass('TOBJECT','EVariantSymbolTable');
15229:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15230:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15231:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15232:   +' : Integer; Value : Variant; end');
15233:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15234:   SIRegister_TVariantSymbolTable(CL);
15235: end;
15236:
15237: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15238: begin
15239:   SIRegister_TThreadLocalVariables(CL);
15240:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD ) : PChar';
15241:   //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD ) : PISC_QUAD';
15242:   Function ThreadLocals : TThreadLocalVariables';
15243:   Procedure WriteDebug( sz : String );
15244:   CL.AddConstantN('UDF_SUCCESS','LongInt'( 0 );
15245:   'UDF_FAILURE','LongInt'( 1 );
15246:   'cSignificantlyLarger','LongInt'( 1024 * 4 );
15247:   CL.AddTypeS('mTByteArray', 'array of byte;');
15248:   function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15249:   function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15250:   procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTTarget: string);
15251:   function IsNetworkConnected: Boolean;
15252:   function IsInternetConnected: Boolean;
15253:   function IsCOMConnected: Boolean;
15254:   function IsNetworkOn: Boolean;
15255:   function IsInternetOn: Boolean;
15256:   function IsCOMON: Boolean;
15257:   Function SetTimer(hWnd : HWND; nIDEEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15258:   TmrProc', 'procedure TmrProc(hWnd : HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);';
15259:   Function SetTimer2( hWnd : HWND; nIDEEvent, uElapse : UINT; lpTimerFunc : TmrProc ) : UINT';
15260:   Function KillTimer( hWnd : HWND; uIDEEvent : UINT ) : BOOL';
15261:   Function wIsWindowUnicode( hWnd : HWND ) : BOOL';
15262:   Function wEnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL';
15263:   Function wIsWindowEnabled( hWnd : HWND ) : BOOL';
15264:   Function GetMenu( hWnd : HWND ) : HMENU';
15265:   Function SetMenu( hWnd : HWND; hMenu : HMENU ) : BOOL';
15266: end;
15267:
15268: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15269: begin
15270:   SIRegister_IDataBlock(CL);
15271:   SIRegister_ISendDataBlock(CL);
15272:   SIRegister_ITransport(CL);
15273:   SIRegister_TDataBlock(CL);
15274:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');

```

```

15275: //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15276: CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15277: CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15278: SIRegister_TCustomDataBlockInterpreter(CL);
15279: SIRegister_TSendDataBlock(CL);
15280: 'CallSig','LongWord') ( $D800);
15281: 'ResultSig','LongWord') ( $D400);
15282: 'asMask','LongWord') ( $00FF);
15283: CL.AddClassN(CL.FindClass('TOBJECT'), 'EInterpreterError');
15284: CL.AddClassN(CL.FindClass('TOBJECT'), 'ESocketConnectionError');
15285: Procedure CheckSignature( Sig : Integer );
15286: end;
15287:
15288: procedure SIRegister_WinInet(CL: TPSPPascalCompiler);
15289: begin
15290: //CL.AddTypeS('HINTERNET', '__Pointer');
15291: CL.AddTypeS('HINTERNET1', 'THANDLE');
15292: CL.AddTypeS('HINTERNET', 'Integer');
15293: CL.AddTypeS('HINTERNET2', '__Pointer');
15294: //CL.AddTypeS('PHINTERNET', 'HINTERNET // will not work');
15295: //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15296: CL.AddTypeS('INTERNET_PORT', 'Word');
15297: //CL.AddTypeS('PININTERNET_PORT', 'INTERNET_PORT // will not work');
15298: //CL.AddTypeS('LPINTERNET_PORT', 'PININTERNET_PORT');
15299: Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15300: 'INTERNET_RFC1123_FORMAT','LongInt'( 0 );
15301: 'INTERNET_RFC1123_BUFSIZE','LongInt'( 30 );
15302: Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
lpUrlComponents:TURLComponents):BOOL;
15303: Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
dwUrlLength:DWORD):BOOL;
15304: Function InternetCloseHandle(hInet : HINTERNET) : BOOL';
15305: Function
InternetConnect (hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15306: Function InternetOpenUrl (hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15307: ;dwContext:DWORD):HINTERNET;
15308: Function InternetOpen (lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15309: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumOfBytesAvail:DWORD;dwFlags,
dwContext:DWORD):BOOL;
15310: Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL';
15311: Function
InternetDial (hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15312: Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD';
15313: Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL';
15314: Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL';
15315: Function InternetAutodialHangup( dwReserved : DWORD) : BOOL';
15316: Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL';
15317: Function InternetCanonicalizeUrl (lpszUrl:PChar;lpszBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15318: Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL';
15319: Function InternetConnect (hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15320: ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15321: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumOfBytesAvail:DWORD;dwFlgs,
dwContext:DWORD):BOOL;
15322: Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET';
15323: Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar;
fFailIfExists:BOOL;dwFlagsAndAttributes: DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL';
15324: Function
WFtpPutFile (hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15325: Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL';
15326: Function FtpRenameFile (hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15327: Function
FtpOpenFile (hConnect:HINTER; lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15328: Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL';
15329: Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL';
15330: Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL';
15331: Function FtpGetCurrentDirectory(hConnect:HINTER; lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15332: Function
FtpCommand (hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15333: Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL';
15334: Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL';
15335: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL';
15336: Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL';
15337: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL';
15338: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL';
15339: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL';
15340: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL';
15341: Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL';
15342: Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL';
15343: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL';
15344: Function
GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorStr:PChar;dwGopherTyp
lpdwBufferLength:DWORD):BOOL;
15345: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL';
15346: Function
GopherOpenFile (hConec:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD) : HINTERNET;

```

```

15347: Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
15348:   PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15349: Function HttpAddRequestHeaders( hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15350: Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15351: Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15352: Function InternetCheckConnection( lpszUrl:PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL';
15353: Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15354: Function InternetConfirmZoneCrossing(hWnd: HWND; szUrlPrev,szUrlNew: LPSTR; bPost : BOOL) : DWORD');
15355: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15356: Function DeleteUrlCacheGroup(GroupID: Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15357: Function FindFirstUrlCacheEntry(lpszUriSearchPattern PChar;var
lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15358: Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpdwNextCacheEntryInfo:TInternetCacheEntryInfo; var
lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15359: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15360: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15361: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15362: Function InternetSetDialState(lpszConnectoid:PChar; dwState:DWORD;dwReserved: DWORD): BOOL');
15363: end;
15364:
15365: procedure SIRegister_Wwstr(CL: TPPSPascalCompiler);
15366: begin
15367:   AddTypeS('str CharSet', 'set of char');
15368:   TwwGetWordOption,'(wwgwSkipLeadingBlanks, wggwQuotesAsWords, wggwStripQuotes , wggwSpacesInWords);
15369:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15370:   Procedure strBreakApart( s : string; delimiter : string; parts : TStrings)');
15371:   Function strGetToken( s : string; delimiter : string; var APos: integer) : string');
15372:   Procedure strStripPreceding( var s : string; delimiter : str CharSet)');
15373:   Procedure strStripTrailing( var s : string; delimiter : str CharSet)');
15374:   Procedure strStripWhiteSpace( var s : string)');
15375:   Function strRemoveChar( str : string; removeChar : char) : string');
15376:   Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15377:   Function strReplaceCharWithStr(str:string; removeChar : char; replaceStr : string) : string');
15378:   Function wwEqualStr( s1, s2 : string) : boolean');
15379:   Function strCount( s : string; delimiter : char) : integer');
15380:   Function strWhiteSpace : str CharSet');
15381:   Function wwExtractFileNameOnly( const FileName : string) : string');
15382:   Function wwGetWord(s:string;var APos:integer;Options:TwwGetWordOptions;DelimSet:str CharSet):string;
15383:   Function strTrailing( s : string; delimiter : char) : string');
15384:   Function strPreceding( s : string; delimiter : char) : string');
15385:   Function wwstrReplace( s, Find, Replace : string) : string');
15386: end;
15387:
15388: procedure SIRegister_DataBkr(CL: TPPSPascalCompiler);
15389: begin
15390:   SIRegister_TRemoteDataModule(CL);
15391:   SIRegister_TCRemoteDataModule(CL);
15392:   Procedure RegisterPooled(const ClassID:string; Max,Timeout : Integer; Singleton : Boolean)');
15393:   Procedure UnregisterPooled( const ClassID : string)');
15394:   Procedure EnableSocketTransport( const ClassID : string)');
15395:   Procedure DisableSocketTransport( const ClassID : string)');
15396:   Procedure EnableWebTransport( const ClassID : string)');
15397:   Procedure DisableWebTransport( const ClassID : string)');
15398: end;
15399:
15400: procedure SIRegister_Mathbox(CL: TPPSPascalCompiler);
15401: begin
15402:   Function mxArcCos( x : Real) : Real');
15403:   Function mxArcSin( x : Real) : Real');
15404:   Function Comp2Str( N : Comp) : String');
15405:   Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15406:   Function Int2Str( N : LongInt) : String');
15407:   Function mxIsEqual( R1, R2 : Double) : Boolean');
15408:   Function LogXY( x, y : Real) : Real');
15409:   Function Pennies2Dollars( C : Comp) : String');
15410:   Function mxPower( X : Integer; Y : Integer) : Real');
15411:   Function Real2Str( N : Real; Width, Places : integer) : String');
15412:   Function mxStr2Comp( MyString : string) : Comp');
15413:   Function mxStr2Pennies( S : String) : Comp');
15414:   Function Str2Real( MyString : string) : Real');
15415:   Function XToThey( x, y : Real) : Real');
15416: end;
15417:
15418: //*****Cindy Functions!*****
15419: procedure SIRegister_cyIndy(CL: TPPSPascalCompiler);
15420: begin
15421:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15422:     +'__Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15423:     +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification);
15424:   MessagePlainText', 'String 'text/plain');
15425:   CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15426:   MessageAlterText_Html','String 'multipart/alternative');
15427:   MessageHtml_Attach','String 'multipart/mixed');
15428:   MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15429:   MessageAlterText_Html_Attach','String 'multipart/mixed');

```

```

15430: MessageAlterText_Html_RelatedAttach,'String')('multipart/related;type="multipart/alternative"');
15431: MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15432: MessageReadNotification','String')('multipart/report;report-type="disposition-notification"');
15433: Function ForceDecodeHeader( aHeader : String ) : String');
15434: Function Base64_EncodeString( Value : String; const aEncoding : TEncoding ) : string');
15435: Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding ) : String;');
15436: Function Base64_DecodeToBytes( Value : String ) : TBytes;');
15437: Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15438: Function Get_MD5( const aFileName : string ) : string');
15439: Function Get_MD5FromString( const aString : string ) : string');
15440: end;
15441:
15442: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15443: begin
15444:   Function IsFolder( SRec : TSearchrec ) : Boolean');
15445:   Function isFolderReadOnly( Directory : String ) : Boolean');
15446:   Function DirectoryIsEmpty( Directory : String ) : Boolean');
15447:   Function DirectoryWithSubDir( Directory : String ) : Boolean');
15448:   Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15449:   Function DiskFreeBytes( Drv : Char ) : Int64');
15450:   Function DiskBytes( Drv : Char ) : Int64');
15451:   Function GetFileBytes( File : String ) : Int64');
15452:   Function GetFilesBytes( Directory, Filter : String ) : Int64');
15453: SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15454: SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15455: SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15456: SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15457: SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15458: SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15459: SE_TCB_NAME','String 'SeTcbPrivilege');
15460: SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15461: SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15462: SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15463: SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15464: SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15465: SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15466: SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15467: SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15468: SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15469: SE_BACKUP_NAME','String 'SeBackupPrivilege');
15470: SE_RESTORE_NAME','String 'SeRestorePrivilege');
15471: SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15472: SE_DEBUG_NAME','String 'SeDebugPrivilege');
15473: SE_AUDIT_NAME','String 'SeAuditPrivilege');
15474: SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15475: SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15476: SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15477: SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15478: SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15479: SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15480: SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15481: end;
15482:
15483: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15484: begin
15485:   Type (TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15486:     +'Me, wvWinNT3, wvWinNT4,wvWin2000,wvWinXP,wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15487:   Function ShellGetExtensionName( FileName : String ) : String');
15488:   Function ShellGetIconIndex( FileName : String ) : Integer');
15489:   Function ShellGetIconHandle( FileName : String ) : HIcon');
15490:   Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15491:   Procedure ShellThreadMove( App_Handle : THandle; fromfile : string; toFile : string)');
15492:   Procedure ShellRenameDir( DirFrom, DirTo : string );
15493:   Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15494:   Procedure cyShellExecute1(ExeFilename,Parameters,AppName,ApplicationClass:String;Restore:Bool);
15495:   Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String );
15496:   Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string );
15497:   Function ShellExecuteEx(FileName: string; const Parameters:string; const Directory: string; const
WaitCloseCompletion : boolean) : Boolean';
15498:   Procedure RestoreAndSetForegroundWindow( Hnd : Integer );
15499:   Function RemoveDuplicatedPathDelimiter( Str : String ) : String');
15500:   Function cyFileTimeToDate( _FT : TFileTime ) : TDateTime');
15501:   Function GetModificationDate( Filename : String ) : TDateTime');
15502:   Function GetCreationDate( Filename : String ) : TDateTime');
15503:   Function GetLastAccessDate( Filename : String ) : TDateTime');
15504:   Function FileDelete( Filename : String ) : Boolean');
15505:   Function FileIsOpen( Filename : string ) : boolean');
15506:   Procedure FilesDelete( FromDirectory : String; Filter : ShortString );
15507:   Function DirectoryDelete( Directory : String ) : Boolean';
15508:   Function GetPrinters( PrintersList : TStrings ) : Integer');
15509:   Procedure SetDefaultPrinter( PrinterName : String );
15510:   Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer );
15511:   Function WinToDosPath( WinPathName : String ) : String');
15512:   Function DosToWinPath( DosPathName : String ) : String');
15513:   Function cyGetWindowsVersion : TWindowsVersion );
15514:   Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean ) : Boolean');
15515:   Procedure WindowsShutDown( Restart : boolean );
15516:   Procedure CreateShortCut( FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
FileIcon:string;NumIcone:integer );

```

```

15517: Procedure GetWindowsFonts( FontsList : TStrings );
15518: Function GetAvailableFilename( DesiredFileName : String ) : String';
15519: end;
15520:
15521: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15522: begin
15523:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15524:   Type(TStrLocateOptions', 'set of TStringLocateOption');
15525:   Type(TStringRead', '( srFromLeft, srFromRight )');
15526:   Type(TStringReads', 'set of TStringRead');
15527:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15528:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15529:   Type(TWordsOptions', 'set of TWordsOption');
15530:   Type(TCarType', '( ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15531:   Type(TCarTypes', 'set of TCarType');
15532:   //CL.AddTypes('TUnicodeCategories', 'set of TUnicodeCategory');
15533:   CarTypeAlphabetic', 'LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15534:   Function Char_GetType( aChar : Char ) : TCarType';
15535:   Function SubString_Count( Str : String; Separator : Char ) : Integer';
15536:   Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15537:   Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word ) : String';
15538:   Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15539:   Procedure SubString_Add( var Str : String; Separator : Char; Value : String );
15540:   Procedure SubString_Insert( var Str:String;Separator:Char;SubStringIndex:Word;Value:String );
15541:   Procedure SubString_Edit( var Str:String;Separator:Char;SubStringIndex:Word;NewValue:String );
15542:   Function SubString_Remove( var Str:string; Separator:Char; SubStringIndex : Word ) : Boolean';
15543:   Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Int;
15544:   Function SubString_Ribbon(Str:string; Separator:Char;Current: Word; MoveBy:Integer):Integer';
15545:   Function SubString_Ribbon1(Str:string;Separator:Char;Current:String;MoveBy:Integer):String';
15546:   Function String_Quote( Str : String ) : String';
15547:   Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char ) : Char';
15548:   Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15549:   Function String_GetWord( Str : String; StringRead : TStringRead ) : String';
15550:   Function String_GetInteger( Str : String; StringRead : TStringRead ) : String';
15551:   Function StringToInt( Str : String ) : Integer';
15552:   Function String_Uppercase( Str : String; Options : TWordsOptions ) : String';
15553:   Function String_Lowercase( Str : String; Options : TWordsOptions ) : String';
15554:   Function String_Reverse( Str : String ) : String';
15555:   Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Int;
15556:   Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive : TCaseSensitive):Integer';
15557:   Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer ) : String';
15558:   Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15559:   Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
  Between2MustExist : Boolean; CaseSensitive : TCaseSensitive ) : String';
15560:   Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer ) : String';
15561:   Function String_Delete1(Str:String;delStr:String; CaseSensitive:TCaseSensitive) : String';
15562:   Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads ) : String';
15563:   Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer ) : String';
15564:   Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Int):String';
15565:   Function String_End( Str : String; Cars : Word ) : String';
15566:   Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
  AlwaysFindFromBeginning : Boolean ) : String';
15567:   Function String_SubstCar( Str : String; Old, New : Char ) : String';
15568:   Function String_Count(Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer';
15569:   Function String_SameCars(Str1,Str2:String;StopCount_Idifferent:Bool;CaseSensitive:TCaseSensitive):Int;
15570:   Function String_IsNumbers( Str : String ) : Boolean';
15571:   Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer ) : Integer';
15572:   Function StringToCsvCell( aStr : String ) : String';
15573: end;
15574:
15575: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15576: begin
15577:   Function LongDayName( aDate : TDate ) : String';
15578:   Function LongMonthName( aDate : TDate ) : String';
15579:   Function ShortYearOf( aDate : TDate ) : byte';
15580:   Function DateToStrYYYYMMDD( aDate : TDate ) : String';
15581:   Function StrYYYYMMDDToDate( aStr : String ) : TDate';
15582:   Function SecondsToMinutes( Seconds : Integer ) : Double';
15583:   Function MinutesToSeconds( Minutes : Double ) : Integer';
15584:   Function MinutesToHours( Minutes : Integer ) : Double';
15585:   Function HoursToMinutes( Hours : Double ) : Integer';
15586:   Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String ) : TDateTime';
15587:   Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer );
15588:   Function MergeDateWithTime( aDate : TDate; aTime : TDateTime ) : TDateTime';
15589:   Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime ) : Int64';
15590:   Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15591:   Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime ) : Int64';
15592:   Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
  RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean';
15593:   Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15594:   Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime ) : Boolean';
15595: end;
15596:
15597: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15598: begin
15599:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15600:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15601:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');

```

```

15602: Type(TcyLocateOptions', 'set of TcyLocateOption');
15603: Function StringsLocate(aList: TStrings; Value:String; Options:TcyLocateOptions): Integer';
15604: Function StringsLocate1(aList:TStrings; Value: String; ValueKind:TStringsValueKind):Integer';
15605: Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean;SortType:TStringsSortType): Integer';
15606: Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind ');
15607: Procedure StringsSort( aList : TStrings; SortType : TStringsSortType );
15608: Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode';
15609: Function TreeNodeLocateOnLevel(TreeView: TTreeView; OnLevel: Integer; Value : String) : TTreeNode';
15610: Function
TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode';
15611: Function TreeNodeGetParentOnLevel(ChildNode: TTreeNode; ParentLevel : Integer) : TTreeNode';
15612: Procedure TreeNodeCopy (FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
CopySubChildren:Bool';
15613: Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)';
15614: Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,String;NewPattern:String;Flags:TReplaceFlags);
15615: Function GetTopMostControlAtPos(FromControl:TWinControl; aControlPoint : TPoint) : TControl';
15616: Procedure cyCenterControl( aControl : TControl )';
15617: Function GetLastParent( aControl : TControl ) : TWinControl';
15618: Function GetControlBitmap( aControl : TWinControl ) : TBitmap';
15619: Function GetRichEditBitmap( aRichEdit : TRichEdit ) : TBitmap';
15620: end;
15621:
15622: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15623: begin
15624:   Function TablePackTable( Tab : TTable ) : Boolean';
15625:   Function TableRegenIndexes( Tab : TTable ) : Boolean';
15626:   Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean';
15627:   Function TableDeleteRecord( Tab : TTable ) : Boolean';
15628:   Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15629:   Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean';
15630:   Function TableEmptyTable( Tab : TTable ) : Boolean';
15631:   Function TableFindKey( aTable : TTable; Value : String ) : Boolean';
15632:   Procedure TableFindNearest( aTable : TTable; Value : String );
15633:   Function
TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
15634:   Function
TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15635:   Function DateToBDESQDate( aDate : TDate; const DateFormat : String ) : String'';
15636: end;
15637:
15638: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15639: begin
15640:   SIRegister_TcyRunTimeDesign(CL);
15641:   SIRegister_TcyShadowText(CL);
15642:   SIRegister_TcyBgPicture(CL);
15643:   SIRegister_TcyGradient(CL);
15644:   SIRegister_tcyBevel(CL);
15645: //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15646:   SIRegister_tcyBevels(CL);
15647:   SIRegister_TcyImagelistOptions(CL);
15648:   Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)'';
15649: end;
15650:
15651: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15652: begin
15653:   Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
Maxdegrade : Byte;'+  

15654:     SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)'';
15655:   Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15656:   Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromCol,toColor:TColor;MaxDegrad:byte);
15657:   Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)'';
15658:   Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)'';
15659:   Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)'';
15660:   Procedure cyFrame(aCanvas:TCanvas; var InsideRect:TRect;Color:TColor;const Width:Integer);';
15661:   Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
BottomColor : TColor; const Width : Integer; const RoundRect : boolean);';
15662:   Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Boolean;const DrawRight:Boolean;const
DrawBottom:Bool;const RoundRect:bool;
15663:   Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
aState : TButtonState; Focused, Hot : Boolean)'';
15664:   Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
aState : TButtonState; Focused, Hot : Boolean)'';
15665:   Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
TColor; aState : TButtonState; Focused, Hot : Boolean)'';
15666:   Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)'';
15667:   Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
const DarkFrameColor:TColor;const LightFrameColor: TColor; const MarkColor : TColor)'';
15668:   Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)'';
15669:   Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15670:   Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt; );
15671:   Procedure cyDrawText(CanvasHandle:Cardinal;Text:String; var Rect:TRect;TextFormat : LongInt)');

```

```

15672: Function cyCreateFontIndirect( fromFont : TFont; Angle : Double ) : TFont;';
15673: Function cyCreateFontIndirect1(fromFont: TFont;CaptionOrientation:TCaptionOrientation):TFont;
15674: Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
CaptionOrientation : TCaptionOrientation;Alignment:TAlignment; Layout : TTextLayout)'';
15675: Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer; const OnlyCalcFoldLine : Boolean ) : TLineCoord)';
15676: Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone : Integer; const OnlyCalcFoldLine : Boolean ) : TLineCoord)';
15677: Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean)';
15678: Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint ) : boolean)';
15679: Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint ) : boolean)';
15680: Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint ) : boolean)';
15681: Procedure DrawCanvas(Destination:TCanvas;DestRect:TRect;Source:TCanvas;SourceRect: TRect);)';
15682: 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);
15683: 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);
15684: 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);
15685: Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)'';
15686: Function ValidGraphic( aGraphic : TGraphic ) : Boolean)';
15687: Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer ) : TColor)';
15688: Function ColorModify( Color : TColor; incR, incG, incB : Integer ) : TColor)';
15689: Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer ) : TColor)';
15690: Function ColorSetPercentPale( Color : TColor; IncPercent : integer ) : TColor)';
15691: Function MediumColor( Color1, Color2 : TColor ) : TColor)';
15692: Function ClientToScreenRect( aControl : TControl; aControlRect : TRect ) : TRect)';
15693: Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect ) : TRect)';
15694: Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect ) : TRect)';
15695: Procedure InflateRectPercent( var aRect : TRect; withPercent : Double )';
15696: Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double ) : TRect)';
15697: Function GetProportionalRect( fromRect, InsideRect : TRect ) : TRect)';
15698: Function PointInRect( const aPt : TPoint; const aRect : TRect ) : boolean)';
15699: Function PointInEllispe( const aPt : TPoint; const aRect : TRect ) : boolean)';
15700: Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String ) : Integer)';
15701: end;
15702:
15703: procedure SIRегистre_cyTypes(CL: TPSPascalCompiler);
15704: begin
15705:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15706:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15707:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawMonochrome )');
15708:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15709:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15710:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,
15711:     + bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft )');
15712:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15713:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15714:   Type(TDgradOrientation', '( dgdVertical,dgdHorizontal,dgdAngle,dgdRadial,dgdRectangle )');
15715:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15716:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
bmInvertReverseFromColor);
15717:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
rjResizeTopLeft,rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15718:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15719:   cCaptionOrientationWarning', 'String') (Note that text orientation doesnt work with all fonts!)';
15720: end;
15721:
15722: procedure SIRегистre_WinSvc(CL: TPSPascalCompiler);
15723: begin
15724:   Const SERVICES_ACTIVE_DATABASEA', 'String' 'ServicesActive');
15725:   SERVICES_ACTIVE_DATABASEW', 'String' 'ServicesActive';
15726:   Const SERVICES_ACTIVE_DATABASE', 'String')' SERVICES_ACTIVE_DATABASEA');
15727:   Const SERVICES_FAILED_DATABASEA', 'String' 'ServicesFailed');
15728:   Const SERVICES_FAILED_DATABASEW', 'String' 'ServicesFailed';
15729:   Const SERVICES_FAILED_DATABASE', 'String' 'SERVICES_FAILED_DATABASEA');
15730:   Const SC_GROUP_IDENTIFIERA', 'String') . '+');
15731:   Const SC_GROUP_IDENTIFIERW', 'String')' '+');
15732:   Const SC_GROUP_IDENTIFIER', 'string 'SC_GROUP_IDENTIFIERA');
15733:   Const SERVICE_NO_CHANGE', 'LongWord $FFFFFF');
15734:   Const SERVICE_ACTIVE', 'LongWord') ( $00000001);
15735:   Const SERVICE_INACTIVE', 'LongWord $00000002);
15736:   Const SERVICE_CONTROL_STOP', 'LongWord $00000001);
15737:   Const SERVICE_CONTROL_PAUSE', 'LongWord $00000002);
15738:   Const SERVICE_CONTROL_CONTINUE', 'LongWord $00000003);
15739:   Const SERVICE_CONTROL_INTERROGATE', 'LongWord $00000004);
15740:   Const SERVICE_CONTROL_SHUTDOWN', 'LongWord $00000005);
15741:   Const SERVICE_STOPPED', 'LongWord $00000001);
15742:   Const SERVICE_START_PENDING', 'LongWord $00000002);
15743:   Const SERVICE_STOP_PENDING', 'LongWord $00000003);
15744:   Const SERVICE_RUNNING', 'LongWord $00000004);
15745:   Const SERVICE_CONTINUE_PENDING', 'LongWord $00000005);
15746:   Const SERVICE_PAUSE_PENDING', 'LongWord $00000006);
15747:   Const SERVICE_PAUSED', 'LongWord $00000007);

```

```

15748: Const SERVICE_ACCEPT_STOP', 'LongWord $00000001);
15749: Const SERVICE_ACCEPT_PAUSE_CONTINUE', 'LongWord $00000002);
15750: Const SERVICE_ACCEPT_SHUTDOWN', 'LongWord $00000004);
15751: Const SC_MANAGER_CONNECT', 'LongWord $0001);
15752: Const SC_MANAGER_CREATE_SERVICE', 'LongWord $0002;
15753: Const SC_MANAGER_ENUMERATE_SERVICE', 'LongWord $0004);
15754: Const SC_MANAGER_LOCK', 'LongWord $0008);
15755: Const SC_MANAGER_QUERY_LOCK_STATUS', 'LongWord $0010);
15756: Const SC_MANAGER_MODIFY_BOOT_CONFIG', 'LongWord $0020);
15757: Const SERVICE_QUERY_CONFIG', 'LongWord $0001);
15758: Const SERVICE_CHANGE_CONFIG', 'LongWord $0002);
15759: Const SERVICE_QUERY_STATUS', 'LongWord $0004);
15760: Const SERVICE_ENUMERATE_DEPENDENTS', 'LongWord $0008);
15761: Const SERVICE_START', 'LongWord $0010);
15762: Const SERVICE_STOP', 'LongWord $0020);
15763: Const SERVICE_PAUSE_CONTINUE', 'LongWord $0040);
15764: Const SERVICE_INTERROGATE', 'LongWord $0080);
15765: Const SERVICE_USER_DEFINED_CONTROL', 'LongWord $0100);
15766: Const SERVICE_KERNEL_DRIVER', 'LongWord $00000001);
15767: Const SERVICE_FILE_SYSTEM_DRIVER', 'LongWord $00000002);
15768: Const SERVICE_ADAPTER', 'LongWord $00000004);
15769: Const SERVICE_RECOGNIZER_DRIVER', 'LongWord $00000008);
15770: Const SERVICE_WIN32_OWN_PROCESS', 'LongWord $00000010);
15771: Const SERVICE_WIN32_SHARE_PROCESS', 'LongWord $00000020);
15772: Const SERVICE_INTERACTIVE_PROCESS', 'LongWord $00000100);
15773: Const SERVICE_BOOT_START', 'LongWord $00000000);
15774: Const SERVICE_SYSTEM_START', 'LongWord $00000001);
15775: Const SERVICE_AUTO_START', 'LongWord $00000002);
15776: Const SERVICE_DEMAND_START', 'LongWord $00000003);
15777: Const SERVICE_DISABLED', 'LongWord $00000004);
15778: Const SERVICE_ERROR_IGNORE', 'LongWord $00000000);
15779: Const SERVICE_ERROR_NORMAL', 'LongWord $00000001);
15780: Const SERVICE_ERROR_SEVERE', 'LongWord $00000002);
15781: Const SERVICE_ERROR_CRITICAL', 'LongWord $00000003);
15782: CL.AddTypeS('SC_HANDLE', 'THandle');
15783: //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15784: CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15785: Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15786: '+: DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15787: +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15788: Const SERVICE_STATUS', '_SERVICE_STATUS');
15789: Const TServiceStatus', '_SERVICE_STATUS');
15790: CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15791: +'playName : PChar; ServiceStatus : TServiceStatus; end');
15792: ENUM SERVICE STATUSA', '_ENUM_SERVICE_STATUSA');
15793: _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15794: TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15795: TEnumServiceStatus', 'TEnumServiceStatusA');
15796: SC_LOCK', '__Pointer');
15797: _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD;lpLockOner:PChar;dwLockDuration:DWORD;end';
15798: _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15799: QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15800: QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15801: TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15802: //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15803: TQueryServiceLockStatus', '_TQueryServiceLockStatusA');
15804: _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15805: +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15806: +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15807: +'iceStartName : PChar; lpDisplayName : PChar; end');
15808: _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15809: QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15810: QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15811: TQueryServiceConfigA', '_TQueryServiceConfigA');
15812: TQueryServiceConfig', '_TQueryServiceConfigA');
15813: Function CloseServiceHandle( hService:SC_HANDLE ) : BOOL';
15814: Function ControlService(hService:SC_HANDLE;dwControl:DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15815: Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;''
15816: +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE');
15817: Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;''
15818: +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar ) : SC_HANDLE');
15819: Function DeleteService( hService : SC_HANDLE ) : BOOL';
15820: Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
TEnumServiceStatus; cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned: DWORD ) : BOOL';
15821: Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState:DWORD;var lpServices:
TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD ) : BOOL';
15822: Function GetServiceKeyName(hSCManager:SC_HANDLE;p DisplayName,lpServiceName:PChar;var
lpcchBuffer:DWORD):BOOL;
15823: Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
lpcchBuffer:DWORD):BOOL;
15824: Function LockServiceDatabase( hSCManager : SC_HANDLE ) : SC_LOCK');
15825: Function NotifyBootConfigStatus( BootAcceptable : BOOL ) : BOOL';
15826: Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD ) : SC_HANDLE');
15827: Function OpenService( hSCManager:SC_HANDLE;lpServiceNam:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15828: Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus,
cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL';
15829: Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;

```

```

15830: Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15831: Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15832: Function UnlockServiceDatabase( ScLock : SC_LOCK ) : BOOL';
15833: end;
15834:
15835: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15836: begin
15837:   Function SelectDate(Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
15838:     AStartOfWeek : TDayOfWeekName; AWeekends : TDaysOfWeek; AWeekendColor:TColor;
15839:     BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15840:   Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
15841:     DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDateTime;
15842:     Function PopupDate(var Date:TDateTime; Edit:TWinControl;MinDate:TDateTime;MaxDate:TDateTime):Boolean;
15843:   Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):TWinControl;
15844:   Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
15845:     AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15846:   Function CreateNotifyThread(const
15847:     FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15848: end;
15849: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15850: begin
15851:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15852:   CL.AddTypeS('TFileCompressionState','(fcNoCompression,fcDefaultCompression,fcLZNT1Compression)');
15853:   Function NtfsGetCompression2( const FileName : TFileName; var State : Short ) : Boolean';
15854:   Function NtfsGetCompression12( const FileName : TFileName ) : TFileCompressionState';
15855:   Function NtfsSetCompression2( const FileName : TFileName; const State : Short ) : Boolean';
15856:   Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)';
15857:   Procedure NtfsSetDirectoryTreeCompression2(const Directy:string;const State:TFileCompressionState)';
15858:   Procedure NtfsSetDefaultFileCompression2(const Directory: string;const State:TFileCompressionState)';
15859:   Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recurve:Bool);
15860:   Function NtfsSetSparse2( const FileName : string ) : Boolean';
15861:   Function NtfsZeroDataByHandle2(const Handle: THandle; const First, Last : Int64) : Boolean';
15862:   Function NtfsZeroDataByName2(const FileName: string; const First, Last : Int64) : Boolean';
15863:   Function NtfsSparseStreamsSupported2( const Volume : string ) : Boolean';
15864:   Function NtfsGetSparse2( const FileName : string ) : Boolean';
15865:   Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD ) : Boolean';
15866:   Function NtfsSetReparsePoint2(const FileName:string;var ReparseData,Size : Longword) : Bool;
15867:   Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD ) : Boolean';
15868:   Function NtfsReparsePointsSupported2( const Volume : string ) : Boolean';
15869:   Function NtfsFileHasReparsePoint2( const Path : string ) : Boolean';
15870:   Function NtfsIsFolderMountPoint2( const Path : string ) : Boolean';
15871:   Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char ) : Boolean';
15872:   Function NtfsMountVolume2(const Volume:WideChar; const MountPoint : WideString) : Boolean';
15873:   CL.AddTypeS('TObjLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15874:   Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15875:   Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15876:   Function NtfsOpLockBreakAcknowledge2(Handle: THandle; Overlapped : TOverlapped) : Boolean';
15877:   Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15878:   Function NtfsRequestOplock2(Handle:THandle; Kind:TObjLock; Overlapped: TOverlapped):Boolean';
15879:   Function NtfsCreateJunctionPoint2( const Source, Destination: string ):Bool;
15880:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15881:     +'urity,siAlternate,siHardLink,siProperty, siObjectIdentifier,siReparsePoints,siSparseFile)');
15882:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15883:   TInternalFindStreamData,'record FileHandle:THandle;Context:TObject; StreamIds:TStreamIds;end';
15884:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15885:     +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15886:   Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15887:   Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean';
15888:   Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean';
15889:   Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String ) : Boolean';
15890:   Function NtfsCreateHardLinkA2(const LinkFileName,ExistingFileName : AnsiString) : Boolean';
15891:   Function NtfsCreateHardLinkW2(const LinkFileName,ExistingFileName : WideString) : Boolean';
15892:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15893:   Function NtfsGetHardLinkInfo2(const FileName: string;var Info: TNtfsHardLinkInfo) : Boolean';
15894:   Function NtfsFindHardLinks2(const Path:string;const FileIdxHigh,FIdxLow:Card;const List:TStrings):Bool;
15895:   Function NtfsDeleteHardlinks2( const FileName : string ) : Boolean';
15896:   FindClass('TOBJECT'),'EJclFileSummaryError');
15897:   TJclFileSummaryAccess,'( fsaRead, fsaWrite, fsaReadWrite )';
15898:   TJclFileSummaryShare,'( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )';
15899:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15900:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15901:   SIRegister_TJclFileSummary(CL);
15902:   SIRegister_TJclFileSummaryInformation(CL);
15903:   SIRegister_TJclDocSummaryInformation(CL);
15904:   SIRegister_TJclMediaFileSummaryInformation(CL);
15905:   SIRegister_TJclMSISummaryInformation(CL);
15906:   SIRegister_TJclShellSummaryInformation(CL);
15907:   SIRegister_TJclStorageSummaryInformation(CL);
15908:   SIRegister_TJclImageSummaryInformation(CL);
15909:   SIRegister_TJclDisplacedSummaryInformation(CL);
15910:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15911:   SIRegister_TJclMiscSummaryInformation(CL);
15912:   SIRegister_TJclWebViewSummaryInformation(CL);

```

```

15913: SIRegister_TJclMusicSummaryInformation(CL);
15914: SIRegister_TJclDRMSummaryInformation(CL);
15915: SIRegister_TJclVideoSummaryInformation(CL);
15916: SIRegister_TJclAudioSummaryInformation(CL);
15917: SIRegister_TJclControlPanelSummaryInformation(CL);
15918: SIRegister_TJclVolumeSummaryInformation(CL);
15919: SIRegister_TJclShareSummaryInformation(CL);
15920: SIRegister_TJclLinkSummaryInformation(CL);
15921: SIRegister_TJclQuerySummaryInformation(CL);
15922: SIRegister_TJclImageInformation(CL);
15923: SIRegister_TJclJpegSummaryInformation(CL);
15924: end;
15925:
15926: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15927: begin
15928:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15929:   T8087Rounding,'( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )';
15930:   T8087Infinity,'( icProjective, icAffine )';
15931:   T8087Exception,'( emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision,
15932:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15933:   Function Get8087ControlWord : Word');
15934:   Function Get8087Infinity : T8087Infinity');
15935:   Function Get8087Precision : T8087Precision');
15936:   Function Get8087Rounding : T8087Rounding');
15937:   Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15938:   Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15939:   Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15940:   Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15941:   Function Set8087ControlWord( const Control : Word) : Word');
15942:   Function ClearPending8087Exceptions : T8087Exceptions');
15943:   Function GetPending8087Exceptions : T8087Exceptions');
15944:   Function GetMasked8087Exceptions : T8087Exceptions');
15945:   Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15946:   Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15947:   Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15948: end;
15949:
15950: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15951: begin
15952:   Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl );
15953:   Procedure BoxMoveAllItems( SrcList, DstList : TWinControl );
15954:   Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
15955:   Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer );
15956:   Procedure BoxMoveSelected( List : TWinControl; Items : TStrings );
15957:   Procedure BoxSetItem( List : TWinControl; Index : Integer );
15958:   Function BoxGetFirstSelection( List : TWinControl ) : Integer ;
15959:   Function BoxCanDropItem(List:TWinControl; X,Y:Integer; var DragIndex : Integer ) : Boolean );
15960: end;
15961:
15962: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15963: begin
15964:   //CL.AddConstantN('SZ_URLCONTEXT','POLEstr').SetString( 'URL Context');
15965:   //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEstr').SetString( 'AsyncCallee');
15966:   CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15967:   type ULONG', 'Cardinal');
15968:     LPCWSTR', 'PChar');
15969:   CL.AddTypeS('LPWSTR', 'PChar');
15970:   LPSTR', 'PChar');
15971:   TBindVerb', 'ULONG');
15972:   TBindInfoF', 'ULONG');
15973:   TBindF', 'ULONG');
15974:   TBSCF', 'ULONG');
15975:   TBindStatus', 'ULONG');
15976:   TCIPStatus', 'ULONG');
15977:   TBindString', 'ULONG');
15978:   TPIFlags', 'ULONG');
15979:   TOIBdgFlags', 'ULONG');
15980:   TParseAction', 'ULONG');
15981:   TFSUAction', 'ULONG');
15982:   TQueryOption', 'ULONG');
15983:   TPUAF', 'ULONG');
15984:   TSZMFlags', 'ULONG');
15985:   TUrlZone', 'ULONG');
15986:   TUrlTemplate', 'ULONG');
15987:   TZAFlags', 'ULONG');
15988:   TUrlZoneReg', 'ULONG');
15989:   'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15990:   CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15991:   const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15992:   const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15993:   const 'CF_NULL','LongInt').SetInt( 0);
15994:   const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15995:   const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15996:   const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15997:   const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15998:   const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15999:   const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
16000:   const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');

```

```

16001: const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
16002: const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
16003: const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
16004: const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
16005: const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
16006: const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
16007: const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
16008: const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
16009: const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
16010: const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
16011: const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
16012: const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
16013: const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
16014: const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
16015: const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
16016: const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
16017: const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
16018: const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
16019: const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
16020: const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
16021: const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
16022: const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
16023: const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
16024: const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
16025: const 'MK_S_ASYNCNCHRONOUS','LongWord').SetUInt( $000401E8);
16026: const 'S_ASYNCNCHRONOUS','LongWord').SetUInt( $000401E8);
16027: const 'E_PENDING','LongWord').SetUInt( $8000000A);
16028: CL.AddInterface(CL.FindInterface("IUNKNOWN"),IBinding, 'IBinding');
16029: SIRegister_IPersistMoniker(CL);
16030: SIRegister_IBindProtocol(CL);
16031: SIRegister_IBinding(CL);
16032: const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
16033: const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
16034: const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
16035: const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
16036: const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
16037: const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
16038: const 'BINDF_ASYNCNCHRONOUS','LongWord').SetUInt( $00000001);
16039: const 'BINDF_ASYNCNSTORAGE','LongWord').SetUInt( $00000002);
16040: const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
16041: const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
16042: const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
16043: const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
16044: const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16045: const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16046: const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16047: const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
16048: const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
16049: const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
16050: const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
16051: const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
16052: const 'BINDF_FREE_THREADS','LongWord').SetUInt( $00010000);
16053: const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
16054: const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
16055: const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
16056: //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
16057: //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
16058: //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
16059: const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16060: const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16061: const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16062: const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16063: const 'BSCF_AVAILABLEDATALSIZEUNKNOWN','LongWord').SetUInt( $00000010);
16064: const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16065: const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16066: const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16067: const 'BINDSTATUS_BEGINDOWNLOADADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16068: const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADADDATA + 1);
16069: const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16070: const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt(BINDSTATUS_ENDDOWNLOADDATA + 1);
16071: const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt(BINDSTATUS_BEGINDOWNLOADCOMPONENTS+1);
16072: const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt(BINDSTATUS_INSTALLINGCOMPONENTS+ 1);
16073: const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16074: const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16075: const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16076: const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16077: const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt(BINDSTATUS_MIMETYPEAVAILABLE+1);
16078: const 'BINDSTATUS_BEGINSYNCOOPERATION','LongInt').SetInt(BINDSTATUS_CACHEFILENAMEAVAILABLE+1);
16079: const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOOPERATION + 1);
16080: const 'BINDSTATUS_BEGINUPLOADADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16081: const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADADDATA + 1);
16082: const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16083: const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16084: const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16085: const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16086: const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16087: const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16088: const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16089: const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt(BINDSTATUS_LOADINGMIMEHANDLER+1);

```

```

16090: const 'BINDSTATUS_FILTERREPORTMIMETYPE', LongInt).SetInt(BINDSTAT_CONTENTDISPOSITIONATTACH+1);
16091: const 'BINDSTATUS_CLSIDCANINSTANTIATE', 'LongInt').SetInt(BINDSTATUS_FILTERREPORTMIMETYPE+ 1);
16092: const 'BINDSTATUS_IUNKNOWNAVAILABLE', 'LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16093: const 'BINDSTATUS_DIRECTBIND', 'LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16094: const 'BINDSTATUS_RAWMIMETYPE', 'LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16095: const 'BINDSTATUS_PROXYDETECTING', 'LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16096: const 'BINDSTATUS_ACCEPTRANGES', 'LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16097: const 'BINDSTATUS_COOKIE_SENT', 'LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16098: const 'BINDSTATUS_COMPACT_POLICY RECEIVED', 'LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16099: const 'BINDSTATUS_COOKIE_SUPPRESSED', 'LongInt').SetInt(BINDSTATUS_COMPACT_POLICY RECEIVED+1);
16100: const 'BINDSTATUS_COOKIE_STATE_UNKNOWN', 'LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16101: const 'BINDSTATUS_COOKIE_STATE_ACCEPT', 'LongInt').SetInt(BINDSTATUS_COOKIE_STATE_UNKNOWN+ 1);
16102: const 'BINDSTATUS_COOKIE_STATE_REJECT', 'LongInt').SetInt(BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16103: const 'BINDSTATUS_COOKIE_STATE_PROMPT', 'LongInt').SetInt(BINDSTATUS_COOKIE_STATE_REJECT + 1);
16104: const 'BINDSTATUS_COOKIE_STATE_LEASH', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16105: const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE', 'LongInt').SetInt(BINDSTATUS_COOKIE_STATE_LEASH+1);
16106: const 'BINDSTATUS_POLICY_HREF', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16107: const 'BINDSTATUS_P3P_HEADER', 'LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16108: const 'BINDSTATUS_SESSION_COOKIE_RECEIVED', 'LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16109: const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED', 'LongInt').SetInt(BINDSTATUS_SESSION_COOKIE_RECEIVED+1);
16110: const 'BINDSTATUS_SESSION_COOKIES_ALLOWED', 'LongInt').SetInt(BINDSTATUS_PERSISTENT_COOKIE_RECEIVED+1);
16111: const 'BINDSTATUS_CACHECONTROL', 'LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16112: const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME', 'LongInt').SetInt(BINDSTATUS_CACHECONTROL + 1);
16113: const 'BINDSTATUS_MIMETEXTPLAINMISMATCH', 'LongInt').SetInt(BINDSTATUS_CONTENTDISPOSITIONFILENAME+1);
16114: const 'BINDSTATUS_PUBLISHERAVAILABLE', 'LongInt').SetInt(BINDSTATUS_MIMETEXTPLAINMISMATCH+1);
16115: const 'BINDSTATUS_DISPLAYNAMEAVAILABLE', 'LongInt').SetInt(BINDSTATUS_PUBLISHERAVAILABLE + 1);
16116: // PBindInfo, '^TBindInfo // will not work';
16117: {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16118: +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16119: +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16120: +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16121: +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16122: TBindInfo', '_tagBINDINFO');
16123: BINDINFO', '_tagBINDINFO');
16124: _REMSecurity_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16125: +'cryptor : DWORD; bInheritHandle : BOOL; end');
16126: TRemSecurityAttributes', '_REMSecurity_ATTRIBUTES');
16127: REMSECURITY_ATTRIBUTES', '_REMSecurity_ATTRIBUTES');
16128: //PremBindInfo', '^TRemBindInfo // will not work';
16129: {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16130: +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16131: +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16132: +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16133: +'n; dwReserved : DWORD; end');
16134: TRemBindInfo', '_tagRemBINDINFO');
16135: RemBINDINFO', '_tagRemBINDINFO');
16136: //PremFormatEtc', '^TRemFormatEtc // will not work';
16137: tagRemFORMATETC', 'record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16138: TRemFormatEtc', '_tagRemFORMATETC');
16139: RemFORMATETC', '_tagRemFORMATETC');
16140: SIRegister_IBindStatusCallback(CL);
16141: SIRegister_IAuthenticate(CL);
16142: SIRegister_IHttpNegotiate(CL);
16143: SIRegister_IWindowForBindingUI(CL);
16144: const 'CIP_DISK_FULL', 'LongInt').SetInt( 0);
16145: const 'CIP_ACCESS_DENIED', 'LongInt').SetInt( CIP_DISK_FULL + 1);
16146: const 'CIP_NEWER_VERSION_EXISTS', 'LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16147: const 'CIP_OLDER_VERSION_EXISTS', 'LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16148: const 'CIP_NAME_CONFLICT', 'LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16149: const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING', 'LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16150: const 'CIP_EXE_SELF_REGISTRATION_TIMEOUT', LongInt(CIP_TRUST_VERIFICATION_COMPONENT_MISSING+1);
16151: const 'CIP_UNSAFE_TO_ABORT', 'LongInt').SetInt( CIP_EXE_SELF_REGISTRATION_TIMEOUT + 1);
16152: const 'CIP_NEED_REBOOT', 'LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16153: const 'CIP_NEED_REBOOT_UI_PERMISSION', 'LongInt').SetInt( CIP_NEED_REBOOT + 1);
16154: SIRegister_ICodeInstall(CL);
16155: SIRegister_IWInetInfo(CL);
16156: const 'WININETINFO_OPTION_LOCK_HANDLE', 'LongInt').SetInt( 65534);
16157: SIRegister_IHttpSecurity(CL);
16158: SIRegister_IWInetHttpInfo(CL);
16159: SIRegister_IBindHost(CL);
16160: const 'URLOSTRM_USECACHEDCOPY_ONLY', 'LongWord').SetUInt( $00000001);
16161: const 'URLOSTRM_USECACHEDCOPY', 'LongWord').SetUInt( $00000002);
16162: const 'URLOSTRM_GETNEWESTVERSION', 'LongWord').SetUInt( $00000003);
16163: Function URLOpenStream(p1:IUnknown; p2: PChar; p3:DWORD; p4:IBindStatusCallback): HResult';
16164: Function URLOpenPullStream(p1:IUnknown; p2:PChar;p3:DWORD; BSC:IBindStatusCallback):HResult';
16165: Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
: IBindStatusCallback) : HResult';
16166: Function URLDownloadToCacheFile( pl : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
IBindStatusCallback ) : HResult';
16167: Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult';
16168: Function HlinkGoBack( unk : IUnknown ) : HResult';
16169: Function HlinkGoForward( unk : IUnknown ) : HResult';
16170: Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR ) : HResult';
16171: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker ) : HResult';
16172: SIRegister_IInternet(CL);
16173: const 'BINDSTRING_HEADERS', 'LongInt').SetInt( 1);
16174: const 'BINDSTRING_ACCEPT_MIMES', 'LongInt').SetInt( BINDSTRING_HEADERS + 1);
16175: const 'BINDSTRING_EXTRA_URL', 'LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);

```

```

16176: const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16177: const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16178: const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16179: const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16180: const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16181: const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16182: const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16183: const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16184: const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16185: const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16186: const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16187: //POLEStrArray', '^TOLESTRArray // will not work');
16188: SIRegister_IInternetBindInfo(CL);
16189: const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16190: const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16191: const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16192: const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16193: const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16194: const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16195: const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16196: const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16197: const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16198: const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16199: const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16200: //const 'PI_DOCFILECLSIDLOOKUP','','').SetString( PI_CLSIDLOOKUP);
16201: //PProtocolData', '^TProtocolData // will not work');
16202: _tagPROTOCOLDATA', record grfFlags:DWORD; dwState:DWORD; pData:TObject; cbData:ULONG; end');
16203: TProtocolData', '_tagPROTOCOLDATA');
16204: PROTOCOLDATA', '_tagPROTOCOLDATA');
16205: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IIInternetProtocolSink,'IIInternetProtocolSink');
16206: SIRegister_IInternetProtocolRoot(CL);
16207: SIRegister_IInternetProtocol(CL);
16208: SIRegister_IInternetProtocolSink(CL);
16209: const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16210: SIRegister_IInternetSession(CL);
16211: SIRegister_IInternetThreadSwitch(CL);
16212: SIRegister_IInternetPriority(CL);
16213: const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16214: const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16215: const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16216: const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16217: const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16218: const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16219: const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16220: const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16221: const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16222: const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16223: const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16224: const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16225: const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16226: const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16227: const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16228: const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16229: const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16230: const 'PSU_DEFAULT','LongInt').SetInt( 1);
16231: const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16232: const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16233: const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16234: const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16235: const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16236: const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16237: const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16238: const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16239: const 'QUERYUSES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16240: const 'QUERY_IS_CACHED','LongInt').SetInt( QUERYUSES_NETWORK + 1);
16241: const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16242: const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16243: const 'QUERYUSES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16244: const 'QUERY_IS_SECURE','LongInt').SetInt( QUERYUSES_CACHE + 1);
16245: const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16246: SIRegister_IInternetProtocolInfo(CL);
16247: IOInet', 'IInternet');
16248: IOInetBindInfo', 'IIInternetBindInfo');
16249: IOInetProtocolRoot', 'IIInternetProtocolRoot');
16250: IOInetProtocol', 'IIInternetProtocol');
16251: IOInetProtocolSink', 'IIInternetProtocolSink');
16252: IOInetProtocolInfo', 'IIInternetProtocolInfo');
16253: IOInetSession', 'IIInternetSession');
16254: IOInetPriority', 'IIInternetPriority');
16255: IOInetThreadSwitch', 'IIInternetThreadSwitch');
16256: Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD;
pszResult:LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16257: Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD;
pszResult:LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16258: Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult';
16259: Function CoInternetGetProtocolFlags( pwzUrl:LPCWSTR;var dwFlags DWORD;dwReserved:DWORD):HResult;
16260: Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult';
16261: Function CoInternetGetSession(dwSessionMode:DWORD; var piInternetSession:
IIInternetSes;dwReserved:DWORD):HResult;

```

```

16262: Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
16263:   pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16264: Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult:LPWSTR;
16265:   cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16266: Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult:LPWSTR;
16267:   cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult';
16268: Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16269: Function OInetGetProtocolFlags(pwzUrl : LPCWSTR; var dwFlags: DWORD; dwReserved : DWORD) : HResult');
16270: Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
16271:   TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16272: Function OInetGetSession(dwSessionMode:DWORD; var
16273:   piInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16274:   //Function CopyBindInfo( const cbisrc : TBindInfo; var bidest : TBindInfo) : HResult';
16275:   //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16276:   // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ) );
16277:   // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ) );
16278:   //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ) );
16279:   //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ) );
16280:   //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ) );
16281:   const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16282:   SIRegister_IInternetSecurityMgrSite(CL);
16283:   const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16284:   const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16285:   const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16286:   const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16287:   const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16288:   const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16289:   const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16290:   const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16291:   const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16292:   const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16293:   const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16294:   const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16295:   const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16296:   const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16297:   const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16298:   const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16299:   const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16300:   const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16301:   const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16302:   const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16303:   const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16304:   const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16305:   const 'S2M_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16306:   const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16307:   SIRegister_IInternetSecurityManager(CL);
16308:   const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16309:   const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16310:   const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16311:   const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16312:   const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16313:   const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16314:   const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16315:   const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16316:   const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16317:   const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16318:   const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16319:   const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16320:   const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16321:   const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16322:   const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16323:   const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16324:   const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16325:   const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16326:   const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16327:   const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16328:   const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16329:   const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16330:   const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16331:   const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16332:   const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16333:   const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16334:   const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16335:   const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16336:   const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16337:   const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16338:   const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16339:   const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16340:   const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16341:   const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16342:   const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16343:   const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16344:   const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16345:   const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);

```

```

16346: const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16347: const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16348: const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16349: const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16350: const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16351: const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16352: const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16353: const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16354: const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16355: const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16356: const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16357: const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16358: const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16359: const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16360: const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16361: const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16362: const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16363: const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16364: const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16365: const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16366: const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16367: const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16368: const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16369: const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16370: const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00080000);
16371: const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16372: const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16373: const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16374: const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16375: const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16376: const 'URLACTION_INFODELIVERY_NO_Removing_CHANNELS','LongWord').SetUInt( $00001D02);
16377: const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16378: const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16379: const 'URLACTION_INFODELIVERY_NO_Removing_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16380: const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16381: const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16382: const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16383: const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16384: const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16385: const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16386: const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16387: const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16388: const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16389: const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16390: const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16391: const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16392: const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16393: const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16394: const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16395: const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16396: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16397: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16398: const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16399: const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16400: const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16401: const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16402: const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16403: const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16404: const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16405: const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16406: const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16407: const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16408: Function GetUrlPolicyPermissions( dw : DWORD ) : DWORD';
16409: Function SetUrlPolicyPermissions( dw, dw2 : DWORD ) : DWORD';
16410: const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0 );
16411: const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0 );
16412: const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1 );
16413: const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1 );
16414: const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1 );
16415: const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1 );
16416: const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999 );
16417: const 'URLZONE_USER_MIN','LongInt').SetInt( 1000 );
16418: const 'URLZONE_USER_MAX','LongInt').SetInt( 10000 );
16419: const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16420: const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16421: const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16422: const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16423: const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16424: const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16425: const 'MAX_ZONE_PATH','LongInt').SetInt( 260 );
16426: const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200 );
16427: const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16428: const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16429: const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16430: const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16431: const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16432: const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16433: const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16434: const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);

```

```

16435: const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16436: //PZoneAttributes', '^TZoneAttributes // will not work');
16437: _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');
16438: { _ZONEATTRIBUTES = packed record
16439:   cbSize: ULONG;
16440:   szDisplayName: array [0..260 - 1] of WideChar;
16441:   szDescription: array [0..200 - 1] of WideChar;
16442:   szIconPath: array [0..260 - 1] of WideChar;
16443:   dwTemplateMinLevel: DWORD;
16444:   dwTemplateRecommended: DWORD;
16445:   dwTemplateCurrentLevel: DWORD;
16446:   dwFlags: DWORD;
16447: end; }
16448: TZoneAttributes', '_ZONEATTRIBUTES');
16449: ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16450: const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16451: const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16452: const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16453: SIRegister_IInternetZoneManager(CL);
16454: SIRegister_IInternetZoneManagerEx(CL);
16455: const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16456: const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16457: const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16458: const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16459: const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16460: const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16461: const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16462: const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16463: //PCodeBaseHold', '^TCodeBaseHold // will not work');
16464: _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16465: +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16466: TCodeBaseHold', '_tagCODEBASEHOLD');
16467: CODEBASEHOLD', '_tagCODEBASEHOLD');
16468: //PSoftDistInfo', '^TSoftDistInfo // will not work');
16469: _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd';
16470: +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHref : LPWSTR; dwI';
16471: +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS';
16472: +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve';
16473: +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16474: TSoftDistInfo', '_tagSOFTDISTINFO');
16475: SOFTDISTINFO', '_tagSOFTDISTINFO');
16476: SIRegister_ISoftDistExt(CL);
16477: Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16478: Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS, dwAdvertisedVersionLS : DWORD) : HResult');
16479: SIRegister_IDataFilter(CL);
16480: //PProtocolFilterData', '^TProtocolFilterData // will not work');
16481: _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolsSink : ';
16482: +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil';
16483: +'terFlags : DWORD; end');
16484: TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16485: PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16486: //PDataInfo', '^TDataInfo // will not work');
16487: _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL';
16488: +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16489: TDataInfo', '_tagDATAINFO');
16490: DATAINFO', '_tagDATAINFO');
16491: SIRegister_IEncodingFilterFactory(CL);
16492: Function IsLoggingEnabled( pszUrl : PChar) : BOOL';
16493: //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL';
16494: //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL';
16495: //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16496: _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU';
16497: +'rlName:LPSTR;StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16498: THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16499: HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16500: Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL';
16501: end;
16502:
16503: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16504: begin
16505: Procedure reformatMemo( const m : TCustomMemo)');
16506: Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16507: Procedure MoveToTop( memo : TMemo)');
16508: Procedure ScrollToTop( memo : TMemo)');
16509: Function LineNumberClicked( memo : TMemo) : integer');
16510: Function MemoClickedLine( memo : TMemo) : integer');
16511: Function ClickedMemoLine( memo : TMemo) : integer');
16512: Function MemoLineClicked( memo : TMemo) : integer');
16513: Function LinePositionClicked( Memo : TMemo) : integer');
16514: Function ClickedMemoPosition( memo : TMemo) : integer');
16515: Function MemoPositionClicked( memo : TMemo) : integer');
16516: Procedure AdjustGridSize( grid : TDrawGrid)');
16517: Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16518: Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16519: Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16520: Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');

```

```

16521: Procedure sortstrDown( var s : string );
16522: Procedure sortstrUp( var s : string );
16523: Procedure rotatestrleft( var s : string );
16524: Function dffstrtofloatdef( s : string; default : extended ) : extended';
16525: Function deblank( s : string ) : string';
16526: Function IntToBinaryString( const n : integer; MinLength : integer ) : string';
16527: Procedure FreeAndClearListBox( C : TListBox );'
16528: Procedure FreeAndClearMemo( C : TMemo );'
16529: Procedure FreeAndClearStringList( C : TStringList );'
16530: Function dffgetfilesize( f : TSearchrec ) : int64';
16531: end;
16532:
16533: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16534: begin
16535:   CL.AddTypeS('intset', 'set of byte');
16536:   TPoint64', 'record x : int64; y : int64; end');
16537:   Function GetNextPandigital( size : integer; var Digits : array of integer ) : boolean';
16538:   Function IsPolygonal( T : int64; var rank : array of integer ) : boolean';
16539:   Function GeneratePentagon( n : integer ) : integer';
16540:   Function IsPentagon( p : integer ) : boolean';
16541:   Function isSquare( const N : int64 ) : boolean';
16542:   Function isCube( const N : int64 ) : boolean';
16543:   Function isPalindrome( const n : int64 ) : boolean';
16544:   Function isPalindromel( const n : int64; var len : integer ) : boolean';
16545:   Function GetEulerPhi( n : int64 ) : int64';
16546:   Function dffIntPower( a, b : int64 ) : int64';
16547:   Function IntPowerl( a : extended; b : int64 ) : extended';
16548:   Function gcd2( a, b : int64 ) : int64';
16549:   Function GCDMany( A : array of integer ) : integer';
16550:   Function LCMMany( A : array of integer ) : integer';
16551:   Procedure ContinuedFraction( A: array of int64;const wholepart:int;var numerator,denominator:int64 );
16552:   Function dffFactorial( n : int64 ) : int64';
16553:   Function digitcount( n : int64 ) : integer';
16554:   Function nextpermute( var a : array of integer ) : boolean';
16555:   Function convertfloattofractionstring(N:extended;maxdenom:integer;multipleof:boolean):string;
16556:   Function convertStringToDecimal( s : string; var n : extended ) : Boolean';
16557:   Function InttoBinaryStr( nn : integer ) : string';
16558:   Function StrToAngle( const s : string; var angle : extended ) : boolean';
16559:   Function AngleToStr( angle : extended ) : string';
16560:   Function deg2rad( deg : extended ) : extended';
16561:   Function rad2deg( rad : extended ) : extended';
16562:   Function GetLongToMercProjection( const long : extended ) : extended';
16563:   Function GetLatToMercProjection( const Lat : Extended ) : Extended';
16564:   Function GetMercProjectionToLong( const ProjLong : extended ) : extended';
16565:   Function GetMercProjectionToLat( const ProjLat : extended ) : extended';
16566:   SIRegister_TPrimes(CL);
16567:   //RIRegister_TPrimes(CL);
16568: //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16569: CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16570:   Function Random64( const N : Int64 ) : Int64';
16571:   Procedure Randomize64';
16572:   Function Random641 : extended';
16573:   Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16574: end;
16575:
16576: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16577: begin
16578:   TrealPoint', 'record x : extended; y : extended; end');
16579:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16580:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16581:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16582:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16583:   PPResult', '( PPoutside, PPinside, PPVertex, PPEdge, PPError )');
16584:   Function realpoint( x, y : extended ) : TRealPoint';
16585:   Function dist( const p1, p2 : TrealPoint ) : extended';
16586:   Function intdist( const p1, p2 : TPoint ) : integer';
16587:   Function dffline( const p1, p2 : TPoint ) : Tline';
16588:   Function Line1( const p1, p2 : TRealPoint ) : TRealline';
16589:   Function dffCircle( const cx, cy, R : integer ) : TCircle';
16590:   Function Circle1( const cx, cy, R : extended ) : TRealCircle';
16591:   Function GetTheta( const L : TLine ) : extended';
16592:   Function GetTheta1( const p1, p2 : TPoint ) : extended';
16593:   Function GetTheta2( const p1, p2 : TRealPoint ) : extended';
16594:   Procedure Extendline( var L : TLine; dist : integer );
16595:   Procedure Extendline1( var L : TRealLine; dist : extended );
16596:   Function Linesintersect( line1, line2 : TLine ) : boolean';
16597:   Function ExtendedlinesIntersect( Line1, Line2:TLine;const extendlines:bool;var IP:TPoint ):bool;
16598:   Function ExtendedlinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint ):bool;
16599:   Function Intersect( L1, L2 : TLine; var pointonborder:boolean; var IP : TPoint ) : boolean';
16600:   Function PointPerpendicularLine( L : TLine; P : TPoint ) : TLine';
16601:   Function PerpDistance( L : TLine; P : TPoint ) : Integer';
16602:   Function AngledLineFromLine(L: TLine; P:TPoint;Dist : extended; alpha : extended ) : TLine';
16603:   Function
AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordiates:bool):TLine;
16604:   Function PointInPoly( const p : TPoint; Points : array of TPoint ) : PPResult';
16605:   Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
Clockwise:bool):integer;
16606:   Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
const screenCoordinates : boolean; const inflateby : integer );

```

```

16607: Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:bool):bool;
16608: Function DegtoRad( d : extended) : extended');
16609: Function RadtoDeg( r : extended) : extended');
16610: Procedure TranslateLeftTo( var L : TLine; newend : TPoint);';
16611: Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);';
16612: Procedure RotateRightEndBy( var L : TLine; alpha : extended);';
16613: Procedure RotateRightEndTo( var L : TLine; alpha : extended);';
16614: Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);';
16615: Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint ) : boolean;');
16616: Function CircleCircleIntersect1(c1,c2 : TRealCircle; var IP1,Ip2 : TRealPoint ) : boolean;');
16617: Function PointCircleTangentLines(const C:TCircle; const P:TPoint;var L1,L2: TLine) : boolean;');
16618: Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,TL2:TLine):Bool;
16619: end;
16620:
16621:
16622: procedure SIRegister_UAstronomy(CL: TPSPPascalCompiler);
16623: begin
16624:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGallLonLat )');
16625:   TDType', '( ttLocal, ttUT, ttGST, ttLST )';
16626:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end';
16627:   TSunrec', 'record TrueEclLon:extended;
16628:     AppEclLon:extended; AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
16629:     TrueAzAlt:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16630:     TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl : '
16631:       +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16632:       +'arth : extended; Phase : extended; end';
16633:     TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16634:       +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end';
16635:     TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16636:       +ct : TDatetime; LastContact: TDateTime; Magnitude:Extended; MaxeclipseUTime:TDateTime;end';
16637:     TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )';
16638:     TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16639:       +extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16640:       +ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16641:       +jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end';
16642:     TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
16643:       extended; UncorrectedEarthDistance : extended; GeoEclLonLat : TRpoint; CorrectedEarthDistance:extended;
16644:       ApparentRaDecl:TRPoint; end';
16645:     SIRegister_TAstronomy(CL);
16646:   Function AngleToStr( angle : extended) : string';
16647:   Function StrToAngle( s : string; var angle : extended) : boolean';
16648:   Function HoursToStr24( t : extended) : string';
16649:   Function RPoint( x, y : extended) : TRPoint';
16650:   Function getStimenename( t : TDType) : string';
16651: end;
16652: procedure SIRegister_UCardComponentV2(CL: TPSPPascalCompiler);
16653: begin
16654:   TCardValue', 'Integer';
16655:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )';
16656:   TShortSuit', '( cardS, cardD, cardC, cardH )';
16657:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16658:   SIRegister_TCard(CL);
16659:   SIRegister_TDeck(CL);
16660: end;
16661: procedure SIRegister_UTGraphSearch(CL: TPSPPascalCompiler);
16662: begin
16663:   tMethodCall', 'Procedure';
16664:   tVerboseCall', 'Procedure ( s : string)';
16665:   // PTEdge', '^TEdge // will not work';
16666:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16667:     +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end';
16668:   SIRegister_TNode(CL);
16669:   SIRegister_TGraphList(CL);
16670: end;
16671: procedure SIRegister_UParser10(CL: TPSPPascalCompiler);
16672: begin
16673:   ParserFloat', 'extended';
16674:   //PParserFloat', '^ParserFloat // will not work';
16675:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16676:     +ulo, IntDiv,IntDIVZ,integerpower,realpower,square,third,fourth,FuncOneVar,FuncTwoVar )';
16677:   //POperation', '^TOperation // will not work';
16678:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16679:     +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16680:     +'; Token : TDFFToken; end';
16681:   TMathProcedure', 'procedure(AnOperation: TDFFOperation)';
16682:   (CL.FindClass('TOBJECT'),'EMathParserError');
16683:   CL.FindClass('TOBJECT','ESyntaxError');
16684:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16685:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16686:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16687:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16688:   (CL.FindClass('TOBJECT'),'EBadName');
16689:   ('TEXParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16690:   SIRegister_TCustomParser(CL);
16691:   SIRegister_TExParser(CL);

```

```

16692: end;
16693:
16694: function isService: boolean;
16695: begin
16696:   result:= NOT(Application is TApplication);
16697:   {result:= Application is TServiceApplication;};
16698: end;
16699: function isApplication: boolean;
16700: begin
16701:   result:= Application is TApplication;
16702: end;
16703: //SM_REMOTESESSION = $1000
16704: function isTerminalSession: boolean;
16705: begin
16706:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16707: end;
16708: writeln(inttostr(getmemoryload))
16709:
16710: procedure SIRегистер_cyIEUtils(CL: TPSPascalCompiler);
16711: begin
16712:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16713:     +String; margin_bottom : String; margin_left : String; margin_right : String'
16714:     +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16715:   Function cyURLEncode( const S : string) : string');
16716:   Function MakeResourceURL( const ModulName:string;const ResName:PChar;const ResType:PChar):string;
16717:   Function MakeResourceURL1( const Modul:HMODULE;const ResName:PChar;const ResType:PChar):string;
16718:   Function cyColorToHtml( aColor : TColor) : String');
16719:   Function HtmlToColor( aHtmlColor : String) : TColor';
16720:   //Function GetStreamEncoding( aStream : TStream) : TEncoding';
16721:   // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean';
16722:   Function AddHtmlUnicodePrefix( aHtml : String) : String';
16723:   Function RemoveHtmlUnicodePrefix( aHtml : String) : String';
16724:   Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16725:   Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16726:   CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16727:   CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16728:   CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16729:   CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16730:   CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16731:   CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16732: end;
16733:
16734:
16735: procedure SIRегистер_UcomboV2(CL: TPSPascalCompiler);
16736: begin
16737:   CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16738:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16739:     +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16740:     +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16741:     +'inationsRepeat, CombinationsRepeatDown )');
16742:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16743:   SIRегистер_TComboSet(CL);
16744: end;
16745:
16746: procedure SIRегистер_cyBaseComm(CL: TPSPascalCompiler);
16747: begin
16748:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16749:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )';
16750:   TProcOnReceiveCommand', 'Procedure ( Sender: TObject; aCommand: Word; userParam : Integer )';
16751:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16752:     +'mmHandle : THandle; aString : String; userParam : Integer )';
16753:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16754:     +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer )';
16755:   SIRегистер_TcyBaseComm(CL);
16756:   CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16757:   CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16758:   Function ValidateFileMappingName( aName : String) : String';
16759:   procedure makeCaption(leftSide, Rightside:string; form:TForm);
16760: end;
16761:
16762: procedure SIRегистер_cyDERUtils(CL: TPSPascalCompiler);
16763: begin
16764:   CL.AddTypeS('DERString', 'String');
16765:   CL.AddTypeS('DERChar', 'Char');
16766:   CL.AddTypeS('TEElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16767:     +'eger, etFloat, etPercentage,etwebSite,etWebMail,etMoney,etDate,etTextLine, etParagraph )');
16768:   CL.AddTypeS('TEElementsTypes', 'set of TEElementsType');
16769:   CL.AddTypeS('DERNString', 'String');
16770:   const DERDecimalSeparator','String').SetString( '.');
16771:   const DERDefaultChars','String')('+@/%-'
16772:   -.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16773:   const DERDefaultChars','String').SetString( '/%-0123456789abcdefghijklmnopqrstuvwxyz');
16774:   Function isValidWebSiteChar( aChar : Char) : Boolean';
16775:   Function isValidWebMailChar( aChar : Char) : Boolean';
16776:   Function isValidwebSite( aStr : String) : Boolean';
16777:   Function isValidWebMail( aStr : String) : Boolean';
16778:   Function ValidateDate( aDERStr: DERString; var RsltFormat : String) : Boolean';
16779:   Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16780:   Function IsDERChar( aChar : Char) : Boolean');

```

```

16780: Function IsDERDefaultChar( aChar : Char ) : Boolean';
16781: Function IsDERMoneyChar( aChar : Char ) : Boolean';
16782: Function IsDERExceptionCar( aChar : Char ) : Boolean';
16783: Function IsDERSymbols( aDERString : String ) : Boolean';
16784: Function StringToDERCharSet( aStr : String ) : DERString';
16785: Function IsDERNDefaultChar( aChar : Char ) : Boolean';
16786: Function IsDERNChar( aChar : Char ) : Boolean';
16787: Function DERToDERNCharset( aDERStr : DERString ) : DERNSString';
16788: Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean ) : String';
16789: Function DERExtractWebMail( aDERStr : DERString ) : String';
16790: Function DERExtractPhoneNr( aDERStr : DERString ) : String';
16791: Function DERExecute( aDERStr:DERString;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean ) :TElementsType;
16792: Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean ) :TElementsType;
16793: Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
aElementsType : TElementsType ) : String';
16794: Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType );';
16795: end;
16796:
16797: procedure SIRegister_cyImage(CL: TPSPPascalCompiler);
16798: begin
16799:   pRGBQuadArray', '^TRGBQuadArray // will not work)';
16800:   Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Int';
16801:   Procedure BitmapSetPercentBrightness(Bmp: TBitmap; IncPercent:Integer; RefreshBmp : Boolean)';
16802:   Procedure BitmapSetPixelsBrightness(Bmp:TBitmap; IncPixels : Integer; RefreshBmp: Boolean)';
16803:   Procedure BitmapSetPercentContrast(Bmp:TBitmap; IncPercent : Integer; RefreshBmp: Boolean)';
16804:   Procedure BitmapSetPixelsContrast(Bmp: TBitmap; IncPixels : Integer; RefreshBmp: Boolean)';
16805:   Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)';
16806:   Procedure BitmapModifyRGB( Bmp : TBitmap;IncRed:Int;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16807:   Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);';
16808:   Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool);';
16809:   Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
SingleDestinationColor, RefreshBmp : Boolean);';
16810:   Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp:Bool)' );
16811:   Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)';
16812:   Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16813:   Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16814:   Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Bool;
16815:   Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16816:   Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)' );
16817:   Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)' );
16818: end;
16819:
16820: procedure SIRegister_WaveUtils(CL: TPSPPascalCompiler);
16821: begin
16822:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msSh, msS, msAh,msA ))';
16823:   TPCMChannel', '( cMono, cStereo )';
16824:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )';
16825:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )';
16826:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1',
16827:   +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b',
16828:   +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b',
16829:   +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b',
16830:   +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b',
16831:   +'it48000Hz, Stereo16bit48000Hz )';
16832:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16833:   +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  cbSize: Word; end';
16834:   tWaveFormatEx', 'PWaveFormatEx';
16835:   HMMIO', 'Integer');
16836:   TWaveDeviceFormats', 'set of TPCMFormat';
16837:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds',
16838:   +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )';
16839:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16840:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )';
16841:   TWaveOutOptions', 'set of TWaveOutOption';
16842:   TStreamOwnership2', '( soReference, soOwned )';
16843:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )';
16844:   // PRawWave', '^TRawWave // will not work)';
16845:   TRawWave', 'record pData : TObject; dwSize : DWORD; end';
16846:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioError');
16847:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioSysError');
16848:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioInvalidOperation');
16849:   TWaveAudioEvent', 'Procedure ( Sender : TObject )';
16850:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW',
16851:   +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)';
16852:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf',
16853:   +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD ) : DWORD';
16854:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu',
16855:   +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean ) : DWORD';
16856:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const ',
16857:   +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)';
16858:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer )';
16859:   TWaveAudioFilterEvent', 'Procedure ( Sender : TObject; const Buffer:TObject; BufferSize:DWORD );

```

```

16860: GetWaveAudioInfo (mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16861: CreateWaveAudio (mmIO:HMMIO;const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16862: CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo );
16863: GetStreamWaveAudioInfo(Stream:TStream;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16864: CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo ):HMMIO;
16865: OpenStreamWaveAudio( Stream : TStream ) : HMMIO';
16866: CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD ) : DWORD';
16867: GetAudioFormat( FormatTag : Word ) : String';
16868: GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx ) : String';
16869: GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD ) : DWORD';
16870: GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx ) : DWORD';
16871: GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx: Integer');
16872: InvertWaveAudio(const Data:TObject;DataSize:DWORD;const pWaveFormat: PWaveFormatEx: Boolean');
16873: SilenceWaveAudio(const Data : TObject; DataSize : DWORD;const pWaveFormat: PWaveFormatEx: Boolean');
16874: ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16875: MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer : TObject; BufferSize : DWORD ); Boolean';
16876: ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD ) : Boolean';
16877: SetPCMFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec : TPCMSamplesPerSec; BitsPerSample : TPCMBytesPerSample ');
16878: Procedure SetPCMFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat );
16879: GetPCMFormat( const pWaveFormat : PWaveFormatEx ) : TPCMFormat';
16880: GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD ) : DWORD';
16881: MS2str( Milliseconds : DWORD; Fmt : TMS2StrFormat ) : String';
16882: WaitForSyncObject( SyncObject : THandle; Timeout : DWORD ) : DWORD';
16883: //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD ) : LRESULT';
16884: end;
16885:
16886: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16887: begin
16888:   'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096 );
16889:   CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000 );
16890:   'PIPE_NAMING_SCHEME','String').SetString( '\%s\pipe\%s' );
16891:   'WAIT_ERROR','LongWord').SetUInt( DWORD( $FFFFFF ) );
16892:   'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1 );
16893:   'STATUS_SUCCESS','LongWord').SetUInt( $00000000 );
16894:   'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005 );
16895:   CL.AddTypeS('TPipeDirection', '( pdi_Duplex, pdi_ClientToServer, pdir_ServerToClient ) ');
16896:   CL.AddTypeS('TPipeType', '( ptyp_BytByte, ptyp_MsgByte, ptyp_MsgMsg ) ');
16897:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete ) ');
16898:   SIRegister_TNamedPipe(CL);
16899:   SIRegister_TSserverPipe(CL);
16900:   SIRegister_TCClientPipe(CL);
16901:   Function CalculateTimeout( aBasis : DWORD ) : DWORD';
16902:   Function HasOverlappedIoCompleted( const ov : OVERLAPPED ) : Boolean';
16903:   Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean): OverlappedResult;
16904:   Function GetStreamAsText( stm : TStream ) : string';
16905:   Procedure SetStreamAsText( const aTxt : string; stm : TStream );
16906: end;
16907:
16908: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16909: begin
16910:   // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16911:   SIRegister_TThumbData(CL);
16912:   'PIC_BMP','LongInt').SetInt( 0 );
16913:   'PIC_JPG','LongInt').SetInt( 1 );
16914:   'THUMB_WIDTH','LongInt').SetInt( 60 );
16915:   'THUMB_HEIGHT','LongInt').SetInt( 60 );
16916:   Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime) : Boolean';
16917:   Procedure GetFileInfo(Filename:string; var Size : Integer; var LastWriteTime : TDateTime );
16918:   Function ReadBitmap( Filehandle : Integer; Width, Height : Integer ) : TBitmap';
16919:   Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap );
16920:   Function OpenPicture( fn : string; var tp : Integer ) : Integer';
16921:   Function ConvertPicture( pi : Integer; tp : Integer ) : TBitmap';
16922:   Function LoadPicture( fn : string; var w, h : Integer ) : TBitmap';
16923:   Function TurnBitmap( bmp : TBitmap; ang : Integer ) : TBitmap';
16924:   Function RotateBitmap( Bitmap : TBitmap; Direction : Integer ) : TBitmap';
16925:   Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap ) : TRect';
16926:   Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer ) : TBitmap';
16927:   Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer );
16928:   Procedure FindFiles( path, mask : string; items : TStringList );
16929:   Function LetFileName( s : string ) : string';
16930:   Function LetParentPath( path : string ) : string';
16931:   Function AddBackSlash( path : string ) : string';
16932:   Function CutBackSlash( path : string ) : string';
16933: end;
16934:
16935: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16936: begin
16937:   // 'BYTES','LongInt').SetInt( 1 );
16938:   'KBYTES','LongInt').SetInt( 1024 );
16939:   'DBG_ALIVE','LongWord').SetUInt( Integer( $11BABEL1 ) );
16940:   'DBG_DESTROYING','LongWord').SetUInt( Integer( $44FADE44 ) );
16941:   'DBG_GONE','LongWord').SetUInt( $99AC1D99 );
16942:   'SHELL_NS_MYCOMPUTER','String').SetString( ':{20D04FE0-3AEA-1069-A2D8-08002B30309D}' );
16943:   SIRegister_MakeComServerMethodsPublic(CL);
16944:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application ) ');

```

```

16945: Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD ) : Boolean';
16946: Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)'';
16947: Function TBGetTempFolder : string';
16948: Function TBGetTempFile : string';
16949: Function TBGetModuleFilename : string');
16950: Function FormatModuleVersionInfo( const aFilename : string ) : string');
16951: Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD ) : string');
16952: Function TBGetFileSize( aFile : string; aMultipleOf : Integer ) : Integer');
16953: Function FormatAttribString( aAttr : Integer ) : string');
16954: Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo ) : string');
16955: Function ShellRecycle( aWnd : HWND; aFileOrFolder : string ) : Boolean');
16956: Function IsDebuggerPresent : BOOL');
16957: Function TBNotImplemented : HRESULT');
16958: end;
16959:
16960: procedure SIRRegister_D2_VistaHelperU(CL: TPPascalCompiler);
16961: begin
16962: //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16963: //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16964: CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16965: CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16966: //TDrivesProperty = array['A'..'Z'] of boolean;
16967: Function TBSetSystemTime( DateTime : TDateTime; DOW : word ) : boolean');
16968: Function IsElevated : Boolean');
16969: Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16970: CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16971: Function TrimNetResource( UNC : string ) : string');
16972: Procedure GetFreeDrives( var FreeDrives : TDrivesProperty );
16973: Procedure GetMappedDrives( var MappedDrives : TDrivesProperty );
16974: Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
UserUsage : TPasswordUsage; User : string; Comment : string ) : boolean');
16975: Function UnmapDrive( Drive : char; Force : boolean ) : boolean');
16976: Function TBIsWindowsVista : Boolean');
16977: Procedure SetVistaFonts( const AForm : TForm );
16978: Procedure SetVistaContentFonts( const AFont : TFont );
16979: Function GetProductType( var sType : String ) : Boolean');
16980: Function lstrcmp( lpString1, lpString2 : PChar ) : Integer');
16981: Function lstrcmpi( lpString1, lpString2 : PChar ) : Integer');
16982: Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer ) : PChar');
16983: Function lstrcpy( lpString1, lpString2 : PChar ) : PChar');
16984: Function lstrcat( lpString1, lpString2 : PChar ) : PChar');
16985: Function lstrlen( lpString : PChar ) : Integer');
16986: Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD ) : BOOL');
16987: Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation: __Pointer; TokenInformationLength: DWORD): BOOL');
16988: end;
16989:
16990: procedure SIRRegister_dwsXPlatform(CL: TPPascalCompiler);
16991: begin
16992: 'clineTerminator','Char').SetString( #10 );
16993: 'clineTerminators','String').SetString( #13#10 );
16994: 'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ) );
16995: SIRRegister_TFixedCriticalSection(CL);
16996: SIRRegister_TMultiReadSingleWrite(CL);
16997: Procedure SetDecimalSeparator( c : Char );
16998: Function GetDecimalSeparator : Char );
16999: Function GetDecimalSeparator : Char );
17000: Procedure SetDateSeparator( c : Char );
17001: Function GetDateSeparator : Char );
17002: Procedure setTimeSeparator( c : Char );
17003: Function getTimeSeparator : Char );
17004: Procedure setShortDateFormat( c : string );
17005: Function getShortDateFormat : string );
17006: Procedure setlongDateFormat( c : string );
17007: Function getlongDateFormat : string );
17008: Procedure setShorttimeFormat( c : string );
17009: Function getShorttimeFormat : string );
17010: Procedure setlongtimeFormat( c : string );
17011: Function getlongtimeFormat : string );
17012: //getTimeSeparator
17013: Procedure setThousandSeparator( c : Char );
17014: Function getThousandSeparator : Char );
17015: Procedure setListSeparator( c : Char );
17016: Function getListSeparator : Char );
17017:
17018: 'LOGON32_LOGON_INTERACTIVE','LongInt').SetInt( 2 );
17019: 'LOGON32_LOGON_NETWORK','LongInt').SetInt( 3 );
17020: 'LOGON32_LOGON_BATCH','LongInt').SetInt( 4 );
17021: 'LOGON32_LOGON_SERVICE','LongInt').SetInt( 5 );
17022: 'LOGON32_PROVIDER_DEFAULT','LongInt').SetInt( 0 );
17023: 'LOGON32_PROVIDER_WINNT35','LongInt').SetInt( 1 );
17024: 'LOGON32_PROVIDER_WINNT40','LongInt').SetInt( 2 );
17025: 'LOGON32_PROVIDER_WINNT50','LongInt').SetInt( 3 );
17026:
17027: TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean )';
17028: Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent );
17029: CL.AddTypeS('NativeInt', 'Integer');

```

```

17030: //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
17031: CL.AddTypeS('NativeUInt', 'Cardinal');
17032: //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
17033: //CL.AddTypeS('TBytes', 'array of Byte');
17034: CL.AddTypeS('RawByteString', 'UnicodeString');
17035: //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
17036: //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
17037: SIRegister_TPath(CL);
17038: SIRegister_TFile(CL);
17039: SIRegister_TdwsThread(CL);
17040: Function GetSystemMilliseconds : Int64';
17041: Function UTCDateTime : TDateTime';
17042: Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString';
17043: Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer';
17044: Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer';
17045: Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer';
17046: Function UnicodeComparePChars(p1:PChar; n1 : Integer; p2 : PChar; n2 : Integer): Integer';
17047: Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer';
17048: Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString';
17049: Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString';
17050: Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer';
17051: Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean';
17052: Function InterlockedIncrement( var val : Integer) : Integer';
17053: Function InterlockedDecrement( var val : Integer) : Integer';
17054: Procedure FastInterlockedIncrement( var val : Integer)');
17055: Procedure FastInterlockedDecrement( var val : Integer)');
17056: Function InterlockedExchangePointer(var target: __Pointer; val: __Pointer) : __Pointer';
17057: Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
17058: Procedure dwsOutputDebugString( const msg : UnicodeString)');
17059: Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
17060: Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
17061: Procedure VarCopy( out dest : Variant; const src : Variant)');
17062: Function VarToUnicodeStr( const v : Variant) : UnicodeString';
17063: Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString';
17064: Function LoadTextFromStream( aStream : TStream) : UnicodeString';
17065: Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString';
17066: Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
17067: Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle';
17068: Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle';
17069: Procedure CloseFileHandle( hFile : THandle)';
17070: Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean';
17071: Function FileMove( const existing, new : UnicodeString) : Boolean';
17072: Function dwsfileDelete( const fileName : String) : Boolean';
17073: Function FileRename( const oldName, newName : String) : Boolean';
17074: Function dwsfileSize( const name : String) : Int64';
17075: Function dwsFileDateTime( const name : String) : TDateTime';
17076: Function DirectSet8087CW( newValue : Word) : Word';
17077: Function DirectSetMXCSR( newValue : Word) : Word';
17078: Function TtoObject( const T: byte) : TObject';
17079: Function TtoPointer( const T: byte) : __Pointer';
17080: Procedure GetMemForT(var T: byte; Size : integer)';
17081: Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer';
17082: end;
17083:
17084: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
17085: begin
17086: 'IPStrSize','LongInt').SetInt( 15 );
17087: 'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711 );
17088: 'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712 );
17089: 'ADWSBASE','LongInt').SetInt( 9000 );
17090: CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17091: +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17092: SIRegister_EApdSocketException(CL);
17093: TWsMode', ` ( wsClient, wsServer ) ');
17094: TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket );
17095: TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer ) ';
17096: SIRegister_TApdSocket(CL);
17097: end;
17098:
17099: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
17100: begin
17101: SIRegister_TApdCustomComPort(CL);
17102: SIRegister_TApdComPort(CL);
17103: Function ComName( const ComNumber : Word) : ShortString';
17104: Function SearchComPort( const C : TComponent) : TApdCustomComPort';
17105: end;
17106:
17107: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
17108: begin
17109: Function inAddBackslash( const S : String) : String';
17110: Function PathChangeExt( const Filename, Extension : String) : String';
17111: Function PathCharCompare( const S1, S2 : PChar) : Boolean';
17112: Function PathCharIsSlash( const C : Char) : Boolean';
17113: Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean';
17114: Function PathCharLength( const S : String; const Index : Integer) : Integer';
17115: Function inPathCombine( const Dir, Filename : String) : String';
17116: Function PathCompare( const S1, S2 : String) : Integer';
17117: Function PathDrivePartLength( const Filename : String) : Integer';
17118: Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;

```

```

17119: Function inPathExpand( const Filename : String ) : String';
17120: Function PathExtensionPos( const Filename : String ) : Integer';
17121: Function PathExtractDir( const Filename : String ) : String';
17122: Function PathExtractDrive( const Filename : String ) : String';
17123: Function PathExtractExt( const Filename : String ) : String';
17124: Function PathExtractName( const Filename : String ) : String';
17125: Function PathExtractPath( const Filename : String ) : String';
17126: Function PathIsRooted( const Filename : String ) : Boolean';
17127: Function PathLastChar( const S : String ) : PChar';
17128: Function PathLastDelimiter( const Delimiters, S : string ) : Integer';
17129: Function PathLowercase( const S : String ) : String';
17130: Function PathNormalizeSlashes( const S : String ) : String';
17131: Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17132: Function PathPos( Ch : Char; const S : String ) : Integer';
17133: Function PathStartsWith( const S, AStartsWith : String ) : Boolean';
17134: Function PathStrNextChar( const S : PChar ) : PChar';
17135: Function PathStrPrevChar( const Start, Current : PChar ) : PChar';
17136: Function PathStrScan( const S : PChar; const C : Char ) : PChar';
17137: Function inRemoveBackslash( const S : String ) : String';
17138: Function RemoveBackslashUnlessRoot( const S : String ) : String';
17139: Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean );
17140: end;
17141:
17142:
17143: procedure SIRegister_CmnFunc2(CL: TPPSPascalCompiler);
17144: begin
17145: NEWREGSTR_PATH_SETUP', 'String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17146: NEWREGSTR_PATH_EXPLORER', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17147: NEWREGSTR_PATH_SPECIAL_FOLDERS', 'String').SetStr(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17148: NEWREGSTR_PATH_UNINSTALL', 'String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17149: NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME', 'String').SetString( 'DisplayName');
17150: NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE', 'String').SetString( 'UninstallString');
17151: KEY_WOW64_64KEY', 'LongWord').SetUInt( $0100);
17152: //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17153: CL.AddTypeS('TLeadByteSet', 'set of Char');
17154: SIRegister_TOneShotTimer(CL);
17155: CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17156: 'RegViews64Bit', 'LongInt').Value.ts32 := ord(rv64Bit);
17157: Function NewfileExists( const Name : String ) : Boolean';
17158: Function inDirExists( const Name : String ) : Boolean';
17159: Function FileOrDirExists( const Name : String ) : Boolean';
17160: Function IsDirectoryAndNotReparsePoint( const Name : String ) : Boolean';
17161: Function GetIniString(const Section,Key:String; Default:String;const Filename:String):String;
17162: Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17163: Function GetIniBool(const Section,Key:String;const Default:Boolean;const Filename:String):Boolean';
17164: Function IniKeyExists( const Section, Key, Filename : String ) : Boolean';
17165: Function IsIniSectionEmpty( const Section, Filename : String ) : Boolean';
17166: Function SetIniString( const Section, Key, Value, Filename : String ) : Boolean';
17167: Function SetIniInt(const Section,Key:String;const Value:Longint;const Filename:String):Bool;
17168: Function SetIniBool(const Section,Key:String;const Value:Bool;const Filename: String):Bool;
17169: Procedure DeleteIniEntry( const Section, Key, Filename : String );
17170: Procedure DeleteIniSection( const Section, Filename : String );
17171: Function GetEnv( const EnvVar : String ) : String';
17172: Function GetCmdTail : String';
17173: Function GetCmdTailEx( StartIndex : Integer ) : String';
17174: Function NewParamCount : Integer';
17175: Function NewParamStr( Index : Integer ) : string';
17176: Function AddQuotes( const S : String ) : String';
17177: Function RemoveQuotes( const S : String ) : String';
17178: Function inGetShortName( const LongName : String ) : String';
17179: Function inGetWinDir : String';
17180: Function inGetSystemDir : String';
17181: Function GetSysWow64Dir : String';
17182: Function GetSysNativeDir( const IsWin64 : Boolean ) : String';
17183: Function inGetTempDir : String';
17184: Function StringChange( var S : String; const FromStr,ToStr : String ) : Integer';
17185: Function StringChangeEx(var S:String;const FromStr>ToStr:String;const SupportDBCS:Bool):Int;
17186: Function AdjustLength( var S : String; const Res : Cardinal ) : Boolean';
17187: Function UsingWinNT : Boolean';
17188: Function ConvertConstPercentStr( var S : String ) : Boolean';
17189: Function ConvertPercentStr( var S : String ) : Boolean';
17190: Function ConstPos( const Ch : Char; const S : String ) : Integer';
17191: Function SkipPastConst( const S : String; const Start : Integer ) : Integer';
17192: Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String ) : Boolean';
17193: Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String ) : Boolean;
17194: Function RegValueExists( H : HKEY; Name : PChar ) : Boolean';
17195: 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;
17196: Function RegOpenKeyExView(const
RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17197: Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar ) : Longint;
17198: Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17199: Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyNam:PChar):Longint;
17200: Function GetShellFolderPath( const FolderID : Integer ) : String';
17201: Function IsAdminLoggedOn : Boolean';
17202: Function IsPowerUserLoggedOn : Boolean';
17203: Function IsMultiByteString( const S : AnsiString ) : Boolean';
17204: Function FontExists( const FaceName : String ) : Boolean';

```

```

17205: //Procedure FreeAndNil( var Obj');
17206: Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE';
17207: Function GetUILanguage : LANGID';
17208: Function RemoveAccelChar( const S : String) : String';
17209: Function GetTextWidth( const DC : HDC; S : String; const Prefix:Boolean):Integer';
17210: Function AddPeriod( const S : String) : String';
17211: Function GetExceptMessage : String';
17212: Function GetPreferredUIFont : String';
17213: Function IsWildcard( const Pattern : String) : Boolean';
17214: Function WildcardMatch( const Text, Pattern : PChar) : Boolean';
17215: Function IntMax( const A, B : Integer) : Integer';
17216: Function Win32ErrorString( ErrorCode : Integer) : String';
17217: Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17218: Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean';
17219: Function DeleteDirTree( const Dir : String) : Boolean';
17220: Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean';
17221: Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17222: // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17223: Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean';
17224: Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean';
17225: Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean';
17226: Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean';
17227: Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17228: Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean';
17229: Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)';
17230: end;
17231:
17232: procedure SIRegister_CmnFunc(CL: TPSPascalCompiler);
17233: begin
17234:   SIRegister_TWindowDisabler(CL);
17235:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )';
17236:   TMMsgBoxCallbackFunc', 'procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17237:   Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17238:   Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17239:   Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer';
17240:   Function MsgBoxP(const Text,Caption:PChar;const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17241:   Function inMsgBox(const Text,Caption:String;const Typ:TMMsgBoxType;const Buttons:Cardinal):Int;
17242:   Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
Typ:TMMsgBoxType;const Buttons:Cardinal):Integer';
17243:   Procedure ReactivateTopWindow)';
17244:   Procedure SetMessageBoxCaption( const Typ : TMMsgBoxType; const NewCaption : PChar)';
17245:   Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)';
17246:   Procedure SetMessageBoxCallbackFunc( const AFunc:TMMsgBoxCallbackFunc;const AParam:LongInt)';
17247: end;
17248:
17249: procedure SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17250: begin
17251:   SIRegister_TImageGrabber(CL);
17252:   SIRegister_TCaptureDrivers(CL);
17253:   SIRegister_TCaptureDriver(CL);
17254: end;
17255:
17256: procedure SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17257: begin
17258:   Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean';
17259:   Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean';
17260: end;
17261:
17262: procedure SIRegister_RedirFunc(CL: TPSPascalCompiler);
17263: begin
17264:   CL.AddTypeS(TPreviousFsRedirectionState,record DidDisable:Boolean;OldValue: __Pointer;end');
17265:   Function AreFsRedirectionFunctionsAvailable : Boolean';
17266:   Function DisableFsRedirectionIf(const Disable:Bool;var PreviousState:TPreviousFsRedirectionState):Bool;
17267:   Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17268:   Function CreateDirectoryRedir(const DisableFsRedir:Boolean; const Filename: String) : BOOL';
17269:   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;
17270:   Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
FailIfExists : BOOL) : BOOL';
17271:   Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL';
17272:   Function DirExistsRedir( const DisableFsRedir:Boolean; const Filename : String) : Boolean';
17273:   Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean';
17274:   Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
TWin32FindData) : THandle';
17275:   Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD';
17276:   Function GetShortNameRedir( const DisableFsRedir:Boolean; const Filename:String) : String';
17277:   Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
TFileVersionNumbers) : Boolean';
17278:   Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17279:   Function MoveFileRedir(const DisableFsRedir:Bool;const Existingfilename,Newfilename:String):BOOL;
17280:   Function MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilen,Newfilename:String;const
Flags:DWORD):BOOL;
17281:   Function NewFileExistsRedir(const DisableFsRedir:Boolean; const Filename:String) : Boolean';
17282:   Function RemoveDirectoryRedir(const DisableFsRedir:Boolean;const Filename : String) : BOOL';

```

```

17283: Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17284: Function SetNTFSCompressionRedir(const DisableFsRedir:Bool;const FileOrDir:String;Compress:Bool:Bool;
17285: SIRegister_TFileRedir(CL);
17286: SIRegister_TTextFileReaderRedir(CL);
17287: SIRegister_TTextFileWriterRedir(CL);
17288: end;
17289:
17290: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17291: begin
17292: //CL.AddTypeS('LongWord', 'Cardinal');
17293: CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17294: Function Compare64( const N1, N2 : Integer64) : Integer';
17295: Procedure Dec64( var X : Integer64; N : LongWord');
17296: Procedure Dec6464( var X : Integer64; const N : Integer64)');
17297: Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord';
17298: Function Inc64( var X : Integer64; N : LongWord) : Boolean';
17299: Function Inc6464( var X : Integer64; const N : Integer64) : Boolean)';
17300: Function Integer64ToStr( X : Integer64) : String';
17301: Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord';
17302: Function Mul64( var X : Integer64; N : LongWord) : Boolean';
17303: Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)';
17304: Procedure Shr64( var X : Integer64; Count : LongWord)';
17305: Function StrToInteger64( const S : String; var X : Integer64) : Boolean)';
17306: end;
17307:
17308: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17309: begin
17310: //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17311: SIRegister_TSsimpleStringList(CL);
17312: CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle )');
17313: CL.AddTypeS('TDetermineDefaultLanguageResult', '(ddNoMatch,ddMatch,ddMatchLangParameter )');
17314: CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17315: CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17316: // TMD5Digest = array[0..15] of Byte;
17317: // TSHA1Digest = array[0..19] of Byte;
17318: Function CheckForMutexes( Mutexes : String) : Boolean';
17319: Function CreateTempDir : String';
17320: Function DecrementSharedCount(const RegView: TRegView; const Filename : String) : Boolean';
17321: Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)';
17322: //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
TDeleteFileProc; const Param : Pointer) : Boolean';
17323: //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
TDetermineDefaultLanguageResult';
17324: //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17325: Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean';
17326: Function GenerateUniqueName(const DisableFsRedir:Bool;Path:Str;const Extension:String):String;
17327: Function GetComputerNameString : String';
17328: Function GetFileDateTime(const DisableFsRedir:Bool;const Filename:String;var DateTime:TFileTime):Bool;
17329: Function GetMD5OfFile(const DisableFsRedir:Boolean; const Filename : String): TMD5Digest';
17330: Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest';
17331: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest';
17332: Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest';
17333: Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest';
17334: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest';
17335: Function GetRegRootKeyName( const RootKey : HKEY) : String';
17336: Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
TotBytes:Int64):Bool;
17337: Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
TotalBytes: Integer64):Bool;
17338: Function GetUserNamesString : String';
17339: Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Bool;
17340: //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
ResultCode:Integer) : Boolean';
17341: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17342: Procedure InternalError( const Id : String)';
17343: Procedure InternalErrorFmt( const S : String; const Args : array of const)';
17344: Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean';
17345: Function IsProtectedSystemFile(const DisableFsRedir:Boolean;const Filename:String):Boolean';
17346: Function MakePendingFileRenameOperationsChecksum : TMD5Digest';
17347: Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean';
17348: Procedure RaiseFunctionFailedError( const FunctionName : String)';
17349: Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)';
17350: Procedure RefreshEnvironment';
17351: Function ReplaceSystemDirWithSysWow64( const Path : String) : String';
17352: Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String';
17353: Procedure UnregisterFont( const FontName, FontFilename : String)';
17354: Function RestartComputer : Boolean';
17355: Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)';
17356: Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)';
17357: Procedure Win32ErrorMsg( const FunctionName : String)';
17358: Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)';
17359: Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean';
17360: //from Func2
17361: //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean);'

```

```

17362: //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):Str';
17363: Procedure RegisterTypeLibrary( const Filename : String');
17364: //Procedure UnregisterTypeLibrary( const Filename : String');
17365: //Function UnpinShellLink( const Filename : String ) : Boolean');
17366: function getVersionInfoEx3: TOSVersionInfoEx;');
17367: Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17368: procedure InitOLE;');
17369: Function ExpandConst( const S : String ) : String');
17370: Function ExpandConstEx( const S : String; const CustomConsts : array of String ) : String');
17371: Function ExpandConstEx2(const S:String;const CustConsts:array of Str;const DoExpandIndivConst:Bool):Str;
17372: Function ExpandConstIfPrefixed( const S : String ) : String');
17373: Procedure LogWindowsVersion');
17374: Function EvalCheck( const Expression : String ) : Boolean');
17375: end;
17376:
17377: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17378: begin
17379:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TResourceDetails');
17380:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17381:   SIRegister_TResourceModule(CL);
17382:   SIRegister_TResourceDetails(CL);
17383:   SIRegister_TAnsiResourceDetails(CL);
17384:   SIRegister_TUnicodeResourceDetails(CL);
17385: Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass );
17386: Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass );
17387: Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer ) : string');
17388: Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer );
17389: Function ResourceNameToInt( const s : string ) : Integer );
17390: Function CompareDetails( p1, p2 : TObject(Pointer) ) : Integer );
17391: end;
17392:
17393:
17394: procedure SIRegister_TSsimpleComPort(CL: TPSPascalCompiler);
17395: begin
17396:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17397:   with CL.AddClassN(CL.FindClass('TObject'), 'TSimpleComPort') do begin
17398:     RegisterMethod('Constructor Create');
17399:     RegisterMethod('Procedure Free');
17400:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17401:     RegisterMethod('Procedure WriteString( const S : String)');
17402:     RegisterMethod('Procedure ReadString( var S : String)');
17403:   end;
17404:   Ex.: SimpleComPort:= TSsimpleComPort.Create;
17405:   SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17406:   SimpleComPort.WriteString(AsciiChar);
17407: end;
17408:
17409:
17410: procedure SIRegister_Console(CL: TPSPascalCompiler);
17411: begin
17412:   CL.AddConstantN('White', 'LongInt').SetInt( 15 );
17413:   // CL.AddConstantN('Blink', 'LongInt').SetInt( 128 );
17414:   ('conBW40','LongInt').SetInt( 0 );
17415:   ('conCO40','LongInt').SetInt( 1 );
17416:   ('conBW80','LongInt').SetInt( 2 );
17417:   ('conCO80','LongInt').SetInt( 3 );
17418:   ('conMono','LongInt').SetInt( 7 );
17419:   CL.AddConstantN('conFont8x8','LongInt').SetInt( 256 );
17420:   //CL.AddConstantN('C40','','').SetString( CO40 );
17421:   //CL.AddConstantN('C80','','').SetString( CO80 );
17422:   Function conReadKey : Char );
17423:   Function conKeyPressed : Boolean );
17424:   Procedure conGotoXY( X, Y : Smallint );
17425:   Function conWhereX : Integer );
17426:   Function conWhereY : Integer );
17427:   Procedure conTextColor( Color : Byte );
17428:   Function conTextColor1 : Byte );
17429:   Procedure conTextBackground( Color : Byte );
17430:   Function conTextBackground1 : Byte );
17431:   Procedure conTextMode( Mode : Word );
17432:   Procedure conLowVideo );
17433:   Procedure conHighVideo );
17434:   Procedure conNormVideo );
17435:   Procedure conClrScr );
17436:   Procedure conClrEol );
17437:   Procedure conInsLine );
17438:   Procedure conDelLine );
17439:   Procedure conWindow( Left, Top, Right, Bottom : Integer );
17440:   Function conScreenWidth : Smallint );
17441:   Function conScreenHeight : Smallint );
17442:   Function conBufferWidth : Smallint );
17443:   Function conBufferHeight : Smallint );
17444:   procedure InitScreenMode );
17445: end;
17446:
17447: (*-----*)
17448: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17449: begin
17450:   SIRegister_TNoRefCountObject(CL);

```

```

17451: Procedure FreeObjects( List : TFPLList );
17452: Procedure GetMethodList( AObject : TObject; AList : TStrings );
17453: Procedure GetMethodList1( AClass : TClass; AList : TStrings );
17454: end;
17455:
17456: procedure SIRегистer_ToolsUnit(CL: TPSPascalCompiler);
17457: begin
17458:   'MaxDataSet','LongInt').SetInt( 35 );
17459:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17460:   SIRегистer_TDBConnector(CL);
17461:   SIRегистer_TDBBasicsTestSetup(CL);
17462:   SIRегистer_TTestDataLink(CL);
17463:   'testValuesCount','LongInt').SetInt( 25 );
17464:   Procedure InitialiseDBConnector';
17465:   Procedure FreeDBConnector';
17466:   Function DateTimeToTimeString( d : tdatetime ) : string';
17467:   Function TStringToDateTIme( d : String ) : TDateTIme';
17468: end;
17469:
17470: procedure SIRегистer_fpcunit(CL: TPSPascalCompiler);
17471: begin
17472:   SIRегистer_EAssertionFailedError(CL);
17473:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17474:   CL.AddTypeS('TRunMethod', 'Procedure');
17475:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TTestResult');
17476:   SIRегистer_TTest(CL);
17477:   SIRегистer_TAssert(CL);
17478:   SIRегистer_TTestFailure(CL);
17479:   SIRегистer_ITestListener(CL);
17480:   SIRегистer_TTestCase(CL);
17481:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17482:   SIRегистer_TTestSuite(CL);
17483:   SIRегистer_TTestResult(CL);
17484:   Function ComparisonMsg( const aExpected : string; const aActual : string ) : string';
17485: end;
17486:
17487: procedure SIRегистer_cTCPBuffer(CL: TPSPascalCompiler);
17488: begin
17489:   TOBJECT ','ETCPBuffer');
17490:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Integer';
17491:     +r; Head : Integer; Used : Integer; end';
17492:   'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500 );
17493:   'ETHERNET_MTU_1GBT','LongInt').SetInt( 9000 );
17494:   'TCP_BUFFER_DEFAULTMAXSIZE','Longint').SetInt( ETHERNET_MTU_1GBT * 4 );
17495:   'TCP_BUFFER_DEFAULTBUFSIZE','Longint').SetInt( ETHERNET_MTU_100MBIT * 4 );
17496:   Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int,const TCPBufSize:Int;
17497:   Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer ) );
17498:   Procedure TCPBufferPack( var TCPBuf : TTCPBuffer ) );
17499:   Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer );
17500:   Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer );
17501:   Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer );
17502:   Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer ) : Pointer;
17503:   Procedure TCPBufferAddBuf(var TCPBuf: TTCPBuffer; const Buf:string; const Size : Integer );
17504:   Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer ) : Integer;
17505:   Function TCPBufferPeek(var TCPBuf:TTCPBuffer;var Buf: string; const Size:Integer): Integer;
17506:   Function TCPBufferRemove(var TCPBuf:TTCPBuffer;var Buf:string;const Size:Integer): Integer;
17507:   Procedure TCPBufferClear( var TCPBuf : TTCPBuffer );
17508:   Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer ) : Integer;
17509:   Function TCPBufferUsed( const TCPBuf : TTCPBuffer ) : Integer;
17510:   Function TCPBufferEmpty( const TCPBuf : TTCPBuffer ) : Boolean;
17511:   Function TCPBufferAvailable( const TCPBuf : TTCPBuffer ) : Integer;
17512:   Function TCPBufferPtr( const TCPBuf : TTCPBuffer ) : Pointer;
17513:   Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer );
17514: end;
17515:
17516: procedure SIRегистer_Glut(CL: TPSPascalCompiler);
17517: begin
17518:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17519:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17520:   CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3 );
17521:   CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12 );
17522:   CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0 );
17523:   CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5 );
17524:   CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6 );
17525:   Procedure LoadGlut( const dll : String );
17526:   Procedure FreeGlut();
17527: end;
17528:
17529: procedure SIRегистer_LEDBitmaps(CL: TPSPascalCompiler);
17530: begin
17531:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17532:   Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean ) : THandle;
17533: end;
17534:
17535: procedure SIRегистer_SwitchLed(CL: TPSPascalCompiler);
17536: begin
17537:   TLEDColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black );
17538:   TTLedState', '( LedOn, LedOff, LedDisabled );
17539:   SIRегистer_TSwitchLed(CL);

```

```

17540: //Procedure Register');
17541: end;
17542:
17543: procedure SIRегистер_FileClass(CL: TPSPPascalCompiler);
17544: begin
17545:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17546:     +'xisting, fdOpenAlways, fdTruncateExisting )');
17547:   CL.AddTypeS('TFileAccess', '( faRead, faWrite, faReadWrite )');
17548:   CL.AddTypeS('TFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17549:   SIRегистер_TCustomFile(CL);
17550:   SIRегистер_TIFile(CL);
17551:   SIRегистер_TMemoryFile(CL);
17552:   SIRегистер_TTextFileReader(CL);
17553:   SIRегистер_TTextFileWriter(CL);
17554:   SIRегистер_TFileMapping(CL);
17555:   SIRегистер_EFileError(CL);
17556: end;
17557:
17558: procedure SIRегистер_FileUtilsClass(CL: TPSPPascalCompiler);
17559: begin
17560:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17561:     +', ffaDirectory, ffaArchive, ffaAnyFile )');
17562:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17563:   SIRегистер_TFileSearch(CL);
17564: end;
17565:
17566: procedure SIRегистер_uColorFunctions(CL: TPSPPascalCompiler);
17567: begin
17568:   TRGBTypte', 'record RedHex : string; GreenHex : string; BlueHex : '
17569:     + string; Red : integer; Green : integer; Blue : integer; end');
17570:   Function FadeColor( aColor : Longint; aFade : integer) : Tcolor';
17571:   Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17572: end;
17573:
17574: procedure SIRегистер_uSettings(CL: TPSPPascalCompiler);
17575: begin
17576:   Procedure SaveOscSettings');
17577:   Procedure GetOscSettings');
17578: end;
17579:
17580: procedure SIRегистер_cyDebug(CL: TPSPPascalCompiler);
17581: begin
17582:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer )';
17583:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17584:     FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17585:       64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17586:         Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17587:   SIRегистер_TcyDebug(CL);
17588: end;
17589:
17590: (*-----*)
17591: procedure SIRегистер_cyCopyFiles(CL: TPSPPascalCompiler);
17592: begin
17593:   TCopyFileResult', '(cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )';
17594:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )';
17595:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )';
17596:   SIRегистер_TDestinationOptions(CL);
17597:   TProcCustomCopyFileEvent', 'Procedure (Sender: TObject; var CopyFileResult:TCopyFileResult)';
17598:   TProcOnCopyFileProgressEvent, Procedure (Sender:TObject; FileBytes,TransferredBytes:int64;PercentDone:Int64);
17599:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String )';
17600:   SIRегистер_TcyCopyFiles(CL);
17601:   Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
17602:     ResetAttr:boolean) : TCopyFileResult';
17603:   Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
17604:     TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar) : TCopyFileResult';
17605:   Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
17606:     ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;
17607:     FileNotExists: TCopyFileNotExists; SubFolders,ResetAttributes:Boolean):Integer';
17608: end;
17609:
17610: procedure SIRегистер_cySearchFiles(CL: TPSPPascalCompiler);
17611: begin
17612:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17613:   SIRегистер_TcyFileAttributes(CL);
17614:   SIRегистер_TSearchRecInstance(CL);
17615:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17616:   TOptions', 'set of TOption';
17617:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )';
17618:   TProcOnValidateFileEvent Procedure (Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
17619:     Accept:boolean;
17620:   TProcOnValidateDirectoryEvent', 'Procedure (Sender:TObject;Directory:String;var Accept:boolean);
17621:   SIRегистер_TcyCustomSearchFiles(CL);
17622:   SIRегистер_TcySearchFiles(CL);
17623:   Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean';
17624:   Function IscyFolder( aSRec : TSearchrec ) : Boolean';
17625: end;
17626:
17627: procedure SIRегистер_jcontrolutils(CL: TPSPPascalCompiler);
17628: begin

```

```

17624: Function jCountChar( const s : string; ch : char ) : integer';
17625: Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings' );
17626: Function jNormalizeDate(const Value:string;theValue:TDateTime;const theFormat:string):string');
17627: Function jNormalizeTime(const Value:string;theValue:TTime;const theFormat:string) : string');
17628: Function jNormalizeDateTime(const Val:string;theValue:TDateTime;const theFormat:string):string';
17629: Function jNormalizeDateSeparator( const s : string ) : string');
17630: Function jIsValidDateString( const Value : string ) : boolean');
17631: Function jIsValidTimeString( const Value : string ) : boolean');
17632: Function jIsValidDateTimeString( const Value : string ) : boolean');
17633: end;
17634:
17635: procedure SIRегистer_kcMapViewer(CL: TPSpascalCompiler);
17636: begin
17637:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TMapViewer');
17638:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
17639:     +'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
17640:     +', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
17641:     +'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
17642:     +'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17643:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17644:   TRealArea', 'record top: Extended; left:Extended; bottom: Extended; right : Extended; end');
17645:   TIntPoint', 'record X : Int64; Y : Int64; end');
17646:   TkRealPoint', 'record X : Extended; Y : Extended; end');
17647:   TOnBeforeDownloadEvent', 'Procedure( Url : string; str : TStream; var CanHandle : Boolean)');
17648:   TOnAfterDownloadEvent', 'Procedure( Url : string; str : TStream)');
17649:   SIRегистer_TCustomDownloadEngine(CL);
17650:   SIRегистer_TCustomGeolocationEngine(CL);
17651:   SIRегистer_TMapViewer(CL);
17652: end;
17653:
17654: procedure SIRегистer_cparserutils(CL: TPSpascalCompiler);
17655: begin
17656:   (*Function isFunc( name : TNamePart ) : Boolean');*)
17657:   Function isUnnamedFunc( name : TNamepart ) : Boolean';
17658:   Function isPtrToFunc( name : TNamePart ) : Boolean');
17659:   Function isFuncRetFuncPtr( name : TNamePart ) : Boolean');
17660:   Function isPtrToFuncRetFuncPtr( name : TNamePart ) : Boolean');
17661:   Function GetFuncParam( name : TNamePart ) : TNamePart');
17662:   Function isArray( name : TNamePart ) : Boolean');
17663:   Function GetArrayPart( name : TNamePart ) : TNamePart');
17664:   Function GetIdFromPart( name : TNamePart ) : AnsiString');
17665:   Function GetIdPart( name : TNamePart ) : TNamePart');
17666:   Function isNamePartPtrToFunc( part : TNamePart ) : Boolean');
17667:   Function isAnyBlock( part : TNamePart ) : Boolean');*)
17668:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17669:   SIRегистer_TLineBreaker(CL);
17670:   CL.AddTypeS('TNameKind', 'Integer');
17671:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TNamePart');
17672:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17673:   Function SphericalMod( X : Extended ) : Extended');
17674:   Function cSign( Value : Extended ) : Extended');
17675:   Function LimitFloat( const eValue, eMin, eMax : Extended ) : Extended');
17676:   Function AngleToRadians( iAngle : Extended ) : Extended');
17677:   Function RadiansToAngle( iRad : Extended ) : Extended');
17678:   Function Cross180( iLong : Double ) : Boolean');
17679:   Function Mod180( Value : integer ) : Integer');
17680:   Function Mod180Float( Value : Extended ) : Extended');
17681:   Function MulDivFloat( a, b, d : Extended ) : Extended');
17682:   Function LongDiff( iLong1, iLong2 : Double ) : Double');
17683:   Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap );
17684:   Function Bmp_CreateFromPersistent( Source : TPersistent ) : TbitMap );
17685:   Function FixFilePath( const Inpath, CheckPath : string ) : string');
17686:   Function UnFixFilePath( const Inpath, CheckPath : string ) : string');
17687:   Procedure FillStringList( sl : TStringList; const aText : string );
17688: end;
17689:
17690: procedure SIRегистer_LedNumber(CL: TPSpascalCompiler);
17691: begin
17692:   TLedSegmentSize', 'Integer');
17693:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
17694:   SIRегистer_TCustomLEDNumber(CL);
17695:   SIRегистer_TLEDNumber(CL);
17696: end;
17697:
17698: procedure SIRегистer_StStrL(CL: TPSpascalCompiler);
17699: begin
17700:   CL.AddTypeS('LStrRec', 'record AllocSize:Longint;RefCount : Longint; Length : Longint; end');
17701:   CL.AddTypeS('AnsiChar', 'Char');
17702:   CL.AddTypeS('BTable', 'array[0..255] of Byte'); //!!!
17703:   Function HexBL( B : Byte ) : AnsiString');
17704:   Function HexWL( W : Word ) : AnsiString');
17705:   Function HexLL( L : LongInt ) : AnsiString');
17706:   Function HexPtr( P : __Pointer ) : AnsiString');
17707:   Function BinaryBL( B : Byte ) : AnsiString');
17708:   Function BinaryWL( W : Word ) : AnsiString');
17709:   Function BinaryLL( L : LongInt ) : AnsiString');
17710:   Function OctalBL( B : Byte ) : AnsiString');
17711:   Function OctalWL( W : Word ) : AnsiString');
17712:   Function OctalLL( L : LongInt ) : AnsiString');

```

```

17713: Function Str2Int16L( const S : AnsiString; var I : SmallInt ) : Boolean';
17714: Function Str2WordL( const S : AnsiString; var I : Word ) : Boolean';
17715: Function Str2LongL( const S : AnsiString; var I : LongInt ) : Boolean';
17716: Function Str2RealL( const S : AnsiString; var R : Double ) : Boolean';
17717: Function Str2RealL( const S : AnsiString; var R : Real ) : Boolean';
17718: Function Str2ExtL( const S : AnsiString; var R : Extended ) : Boolean';
17719: Function Long2StrL( L : LongInt ) : AnsiString';
17720: Function Real2StrL( R : Double; Width : Byte; Places : ShortInt ) : AnsiString';
17721: Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt ) : AnsiString';
17722: Function ValPrepL( const S : AnsiString ) : AnsiString';
17723: Function CharStrL( C : Char; Len : Cardinal ) : AnsiString';
17724: Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal ) : AnsiString';
17725: Function PadIL( const S : AnsiString; Len : Cardinal ) : AnsiString';
17726: Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal ) : AnsiString';
17727: Function LeftPadL( const S : AnsiString; Len : Cardinal ) : AnsiString';
17728: Function TrimLeadL( const S : AnsiString ) : AnsiString';
17729: Function TrimTrailL( const S : AnsiString ) : AnsiString';
17730: Function TrimL( const S : AnsiString ) : AnsiString';
17731: Function TrimSpacesL( const S : AnsiString ) : AnsiString';
17732: Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal ) : AnsiString';
17733: Function CenterL( const S : AnsiString; Len : Cardinal ) : AnsiString';
17734: Function EntabL( const S : AnsiString; TabSize : Byte ) : AnsiString';
17735: Function DetabL( const S : AnsiString; TabSize : Byte ) : AnsiString';
17736: Function ScrambleL( const S, Key : AnsiString ) : AnsiString';
17737: Function SubstituteL( const S, FromStr, ToStr : AnsiString ) : AnsiString';
17738: Function FilterL( const S, Filters : AnsiString ) : AnsiString';
17739: Function CharExistsL( const S : AnsiString; C : AnsiChar ) : Boolean';
17740: Function CharCountL( const S : AnsiString; C : AnsiChar ) : Cardinal';
17741: Function WordCountL( const S, WordDelims : AnsiString ) : Cardinal';
17742: Function WordPositionL(N:Cardinal;const S,WordDelims:AnsiString;var Pos:Cardinal):Boolean';
17743: Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString ) : AnsiString';
17744: Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar ) : Cardinal';
17745: Function AsciiPositionL(N:Card;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17746: Function ExtractAsciiL(N:Cardinal;const S,WordDelims: AnsiString;Quote: AnsiChar):AnsiString';
17747: Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17748: Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Card;PadToMargin:Bool;
17749: Function CompStringL( const S1, S2 : AnsiString ) : Integer';
17750: Function CompUCStringL( const S1, S2 : AnsiString ) : Integer';
17751: Function SoundexL( const S : Ansistring ) : AnsiString';
17752: Function MakeLetterSetL( const S : Ansistring ) : Longint';
17753: Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable );
17754: Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
Pos:Card):Bool;
17755: Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
var Pos : Cardinal ) : Boolean';
17756: Function DefaultExtensionL( const Name, Ext : AnsiString ) : AnsiString';
17757: Function ForceExtensionL( const Name, Ext : AnsiString ) : AnsiString';
17758: Function JustFilenameL( const PathName : AnsiString ) : AnsiString';
17759: Function JustNameL( const PathName : AnsiString ) : AnsiString';
17760: Function JustExtensionL( const Name : AnsiString ) : AnsiString';
17761: Function JustPathnameL( const PathName : AnsiString ) : AnsiString';
17762: Function AddBackSlashL( const DirName : AnsiString ) : AnsiString';
17763: Function CleanPathNameL( const PathName : AnsiString ) : AnsiString';
17764: Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal ) : Boolean';
17765: Function CommaizeL( L : LongInt ) : AnsiString';
17766: Function CommaizeChL( L : Longint; Ch : AnsiChar ) : AnsiString';
17767: Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr,RtCurr:AnsiString;Sep,
DecPt:Char):AnsiString;
17768: Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
RtCurr:AnsiString;Sep:Char):AnsiString';
17769: Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal ) : Boolean';
17770: Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal ) : Boolean';
17771: Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal ) : AnsiString';
17772: Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal ) : AnsiString';
17773: Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal ) : AnsiString';
17774: Function StrChDeleteL( const S : AnsiString; Pos : Cardinal ) : AnsiString';
17775: Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal ) : AnsiString';
17776: Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal ) : Boolean';
17777: Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal ) : Boolean';
17778: Function CopyLeftL( const S : AnsiString; Len : Cardinal ) : AnsiString';
17779: Function CopyMidL( const S : AnsiString; First, Len : Cardinal ) : AnsiString';
17780: Function CopyRightL( const S : AnsiString; First : Cardinal ) : AnsiString';
17781: Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal ) : AnsiString';
17782: Function CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
SubString:AString):Bool;
17783: Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
SubString:Ansistring):Bool;
17784: Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean ) : AnsiString';
17785: Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
var SubString : AnsiString ) : Boolean';
17786: Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
SubString : AnsiString ) : Boolean';
17787: Function DeleteWithinL( const S, Delimiter : AnsiString ) : AnsiString';
17788: Function ExtractTokensL(const S,
Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Card;
17789: Function IsChAlphaL( C : AnsiChar ) : Boolean';
17790: Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean';
17791: Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString ) : Boolean';
17792: Function IsStrAlphaL( const S : AnsiString ) : Boolean';

```

```

17793: Function IsStrNumericL( const S, Numbers : AnsiString ) : Boolean';
17794: Function IsStrAlphaNumericL( const S, Numbers : AnsiString ) : Boolean';
17795: Function KeepCharsL( const S, Chars : AnsiString ) : AnsiString';
17796: Function LastWordL( const S, WordDelims, AWord:AnsiString; var Position: Cardinal ) : Boolean';
17797: Function LastWordAbsL( const S, WordDelims : AnsiString; var Position: Cardinal ) : Boolean';
17798: Function LastStringL( const S, AString : AnsiString; var Position : Cardinal ) : Boolean';
17799: Function LeftTrimCharsL( const S, Chars : AnsiString ) : AnsiString';
17800: Function ReplaceWordL( const S, WordDelims, OldWord, NewWord:AnsiString; N:Cardinal; var Replacements:Cardinal ):AnsiString;
17801: Function ReplaceWordAllL( const S, WordDelims, OldWord, NewWord:AnsiString; var Replacements:Cardinal ):AnsiString';
17802: Function ReplaceStringL( const S, OldString, NewString:AnsiString; N:Cardinal; var Replacements:Cardinal ):AnsiString;
17803: Function ReplaceStringAllL( const S, OldString, NewString:AnsiString; var Replacements:Cardinal ):AnsiString;
17804: Function RepeatStringL( const RepeatString:AnsiString; var Repetitions:Cardinal; MaxLen:Cardinal ) : AnsiString;
17805: Function RightTrimCharsL( const S, Chars : AnsiString ) : AnsiString';
17806: Function StrWithinL( const S, SearchStr:string; Start:Cardinal; var Position:Cardinal ) : boolean';
17807: Function TrimCharsL( const S, Chars : AnsiString ) : AnsiString';
17808: Function WordPosL( const S, WordDelims, AWord:AnsiString; N:Cardinal; var Position:Cardinal ) : Bool;
17809: Function WordPos( const S, WordDelims, AWord:AnsiString; N:Cardinal; var Position:Cardinal ) : Bool;
17810: end;
17811:
17812: procedure SIRegister_pwnative_out(CL: TPSPPascalCompiler);
17813: begin
17814: CL.AddConstantN('STDIN','LongInt').SetInt( 0 );
17815: ('STDOUT','LongInt').SetInt( 1 );
17816: ('STDERR','LongInt').SetInt( 2 );
17817: Procedure NativeWrite( s : astr );';
17818: Procedure NativeWriteln( PString : PChar );';
17819: Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal );';
17820: Procedure NativeWriteLn( s : astr );';
17821: Procedure NativeWriteLn1();';
17822: end;
17823:
17824: procedure SIRegister_synwrap1(CL: TPSPPascalCompiler);
17825: begin
17826: CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse : integer; UltimateURL : ansistring; Headers : ansistring; end');
17827: CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17828: Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean ) : TSynwInfo;';
17829: Function GetHttpFile1( const Url, UserAgent, Outfile : string ) : TSynwInfo;';
17830: Function GetHttpFile2( const Url, Outfile : string ) : TSynwInfo;';
17831: Function GetHttpFile3( const Url, outfile : string; verbose : boolean ) : TSynwInfo;';
17832: Function GetHtm( const Url : string ) : string;';
17833: Function GetHtml( const Url, UserAgent : string ) : string;';
17834: Function GetUrl( const Url : string; verbose : boolean ) : TSynwInfo;';
17835: Function GetUrl1( const Url, useragent : string ) : TSynwInfo;';
17836: Function GetUrl2( const Url : string ) : TSynwInfo;';
17837: Function GetUrl3( const Url : string; const http : THHTPSend; verbose : boolean ) : TUrlInfo;';
17838: Function GetUrl4( const Url : string; const http : THHTPSend ) : TUrlInfo;';
17839: Procedure StrToStream( s : String; strm : TMemoryStream );
17840: Function StrLoadStream( strm : TStream ) : String';
17841: end;
17842:
17843:
17844: procedure SIRegister_HTMLUtil(CL: TPSPPascalCompiler);
17845: begin
17846: Function GetVal( const tag, attribname_ci : string ) : string';
17847: Function GetTagName( const Tag : string ) : string';
17848: Function GetUpTagName( const tag : string ) : string';
17849: Function GetNameValPair( const tag, attribname_ci : string ) : string';
17850: Function GetValFromNameVal( const namevalpair : string ) : string';
17851: Function GetNameValPair_cs( const tag, attribname : string ) : string';
17852: Function GetVal_JAMES( const tag, attribname_ci : string ) : string';
17853: Function GetNameValPair_JAMES( const tag, attribname_ci : string ) : string';
17854: Function CopyBuffer( StartIndex : PChar; Len : integer ) : string';
17855: Function Ucase( s : string ) : string';
17856: end;
17857:
17858: procedure SIRegister_pwmain(CL: TPSPPascalCompiler);
17859: begin
17860: CL.AddConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT' );
17861: EXPIRED_COOKIE','String').SetString( 'Mon, 01 Jan 2001 12:00:00 GMT' );
17862: 'SECURE_OFF','LongInt').SetInt( 0 );
17863: 'SECURE_ON','LongInt').SetInt( 2 );
17864: 'SECURE_FILTER','LongInt').SetInt( 3 );
17865: THandle or DWord!
17866: // astr = ansistring;
17867: CL.AddTypeS('pastr', 'ansistring');
17868: CL.AddTypeS('TFilterFunc', 'function(const s: pastr): pastr;');
17869: uses pwinit at begin
17870: //type TFilterFunc = function(const s: astr): astr;
17871: Demo: ..\maxbox3\examples2\519_powlts.txt
17872:
17873: //CL.AddConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false );
17874: //CL.AddConstantN('CASE_IGNORE','Boolean')BoolToStr( false );
17875: Procedure pwInit();
17876: Procedure OffReadIn();
17877: Function Lcase( const s : pastr ) : pastr';
17878: Function Ucase( const s : pastr ) : pastr';

```

```

17879: Function CountPostVars : longword';
17880: Function GetPostVar( const name : pastr ) : pastr;';
17881: Function GetPostVar1( const name : pastr; filter : TFilterFunc ) : pastr;');
17882: Function GetPostVar_S( const name : pastr; Security : integer ) : pastr');
17883: Function GetPostVar_SF( const name : pastr; Security : integer ) : pastr');
17884: Function GetPostVarAsFloat( const name : pastr ) : double');
17885: Function GetPostVarAsInt( const name : pastr ) : longint');
17886: Function GetPostVar_SafeHTML( const name : pastr ) : pastr');
17887: Function FetchPostVarName( idx : longword ) : pastr');
17888: Function FetchPostVarVal( idx : longword ) : pastr');
17889: Function FetchPostVarVal1( idx : longword; filter : TFilterFunc ) : pastr');
17890: Function FetchPostVarName_S( idx : longword; Security : integer ) : pastr');
17891: Function FetchPostVarVal_S( idx : longword; Security : integer ) : pastr');
17892: Function IsPostVar( const name : pastr ) : boolean');
17893: Function CountAny : longword');
17894: Function GetAny( const name : pastr ) : pastr');
17895: Function GetAny1( const name : pastr; filter : TFilterFunc ) : pastr');
17896: Function GetAny_S( const name : pastr; Security : integer ) : pastr');
17897: Function GetAnyAsFloat( const name : pastr ) : double');
17898: Function GetAnyAsInt( const name : pastr ) : longint');
17899: Function IsAny( const name : pastr ) : byte');
17900: Function CountCookies : longword');
17901: Function FetchCookieName( idx : longword ) : pastr');
17902: Function FetchCookieVal( idx : longword ) : pastr');
17903: Function FetchCookieVall( idx : longword; filter : TFilterFunc ) : pastr');
17904: Function GetCookie( const name : pastr ) : pastr');
17905: Function GetCookie1( const name : pastr; filter : TFilterFunc ) : pastr');
17906: Function GetCookieAsFloat( const name : pastr ) : double');
17907: Function GetCookieAsInt( const name : pastr ) : longint');
17908: Function IsCookie( const name : pastr ) : boolean');
17909: Function SetCookie( const name, value : pastr ) : boolean');
17910: Function SetCookieAsFloat( const name : pastr; value : double ) : boolean');
17911: Function SetCookieAsInt( const name : pastr; value : longint ) : boolean');
17912: Function SetCookieEx( const name, value, path, domain, expiry : pastr ) : boolean');
17913: Function SetCookieAsFloatEx( const name:pastr;value:double;const path,domai,expiry:pastr):bool;
17914: Function SetCookieAsIntEx(const name:pastr;value:longint;const path,domai,expiry:pastr):bool;
17915: Function UnsetCookie( const name : pastr ) : boolean');
17916: Function UnsetCookieEx( const name, path, domain : pastr ) : boolean');
17917: Function FilterHtml( const input : pastr ) : pastr');
17918: Function FilterHtml_S( const input : pastr; security : integer ) : pastr');
17919: Function TrimBadChars( const input : pastr ) : pastr');
17920: Function TrimBadFile( const input : pastr ) : pastr');
17921: Function TrimBadDir( const input : pastr ) : pastr');
17922: Function TrimBad_S( const input : pastr; security : integer ) : pastr');
17923: Function CountHeaders : longword');
17924: Function FetchHeaderName( idx : longword ) : pastr');
17925: Function FetchHeaderVal( idx : longword ) : pastr');
17926: Function GetHeader( const name : pastr ) : pastr');
17927: Function IsHeader( const name : pastr ) : boolean');
17928: Function SetHeader( const name, value : pastr ) : boolean');
17929: Function UnsetHeader( const name : pastr ) : boolean');
17930: Function PutHeader( const header : pastr ) : boolean');
17931: Procedure Out1( const s : pastr );
17932: Procedure OutLn( const s : pastr );
17933: Procedure OutA( args : array of const );
17934: Procedure OutF( const s : pastr );
17935: Procedure OutLnF( const s : pastr );
17936: Procedure OutFF( const s : pastr );
17937: Procedure OutF_FI( const s : pastr; HTMLFilter : boolean );
17938: Procedure OutLnFF( const s : pastr );
17939: Procedure OutLnF_FI( const s : pastr; HTMLFilter : boolean );
17940: Function FileOut( const fname : pastr ) : word');
17941: Function ResourceOut( const fname : pastr ) : word');
17942: Procedure BufferOut( const buff, len : LongWord );
17943: Function TemplateOut( const fname : pastr; HtmlFilter : boolean ) : word');
17944: Function TemplateOut1( const fname : pastr ) : word');
17945: Function TemplateOut2( const fname : pastr; filter : TFilterFunc ) : word');
17946: Function TemplateOut3( const fname : pastr; HtmlFilter : boolean ) : word');
17947: Function TemplateRaw( const fname : pastr ) : word');
17948: Function Fmt( const s : pastr ) : pastr');
17949: Function Fmt1( const s : pastr; filter : TFilterFunc ) : pastr');
17950: Function FmtFilter( const s : pastr ) : pastr');
17951: Function Fmt_SF( const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecur,TrimSecurity:int ):pastr;
17952: Function Fmt_SF1( const s:pastr;HTMLFilter:bool;FilterSecurity,TrimSecurity:integer ):pastr );
17953: Function CountRtiVars : longword');
17954: Function FetchRtiName( idx : longword ) : pastr');
17955: Function FetchRtiVal( idx : longword ) : pastr');
17956: Function GetRti( const name : pastr ) : pastr');
17957: Function GetRtiAsFloat( const name : pastr ) : double');
17958: Function GetRtiAsInt( const name : pastr ) : longint');
17959: Function IsRti( const name : pastr ) : boolean');
17960: Procedure SetRTI( const name, value : pastr );
17961: Function FetchUpfileName( idx : longword ) : pastr');
17962: Function GetUpfileName( const name : pastr ) : pastr');
17963: Function GetUpfileSize( const name : pastr ) : longint');
17964: Function GetUpFileType( const name : pastr ) : pastr');
17965: Function CountUpfiles : longword');
17966: Function IsUpfile( const name : pastr ) : boolean');
17967: Function SaveUpfile( const name, fname : pastr ) : boolean');

```

```

17968: Function CountVars : longword');
17969: Function FetchVarName( idx : longword) : pastr');
17970: Function FetchVarVal( idx : longword) : pastr;');
17971: Function FetchVarVal1( idx : longword; filter : TFilterFunc) : pastr');
17972: Function GetVar( const name : pastr) : pastr');
17973: Function GetVar1( const name : pastr; filter : TFilterFunc) : pastr');
17974: Function GetVar_S( const name : pastr; security : integer) : pastr');
17975: Function GetVarAsFloat( const name : pastr) : double');
17976: Function GetVarAsInt( const name : pastr) : longint');
17977: Procedure SetVar( const name, value : pastr)');
17978: Procedure SetVarAsFloat( const name : pastr; value : double)');
17979: Procedure SetVarAsInt( const name : pastr; value : longint)');
17980: Function IsVar( const name : pastr) : byte');
17981: Procedure UnsetVar( const name : pastr)');
17982: Function LineEndToBR( const s : pastr) : pastr');
17983: Function RandomStr( len : longint) : pastr');
17984: Function XorCrypt( const s : pastr; key : byte) : pastr');
17985: Function CountCfgVars : longword');
17986: Function FetchCfgVarName( idx : longword) : pastr');
17987: Function FetchCfgVarVal( idx : longword) : pastr');
17988: Function IsCfgVar( const name : pastr) : boolean');
17989: Function SetCfgVar( const name, value : pastr) : boolean');
17990: Function GetCfgVar( const name : pastr) : pastr');
17991: Procedure ThrowErr( const s : pastr)');
17992: Procedure ThrowWarn( const s : pastr)');
17993: Procedure ErrWithHeader( const s : pastr)');
17994: CL.AddTypeS('TWebVar', 'record name : pastr; value : pastr; end');
17995: CL.AddTypeS('TWebVars', 'array of TWebVar');
17996: Function iUpdateWebVar(var webv:TWebVars; const name,value:pastr; upcased:boolean):boolean');
17997: Function iAddWebCfgVar( const name, value : pastr) : boolean');
17998: Procedure iAddWebVar( var webv : TWebVars; const name, value : pastr)');
17999: Procedure iSetRTI( const name, value : pastr)');
18000: Function iCustomSessUnitSet : boolean');
18001: Function iCustomCfgUnitSet : boolean');
18002: end;
18003:
18004: procedure SIRегистre_W32VersionInfo(CL: TPPascalCompiler);
18005: begin
18006:   SIRегистre_TProjectVersionInfo(CL);
18007:   Function MSLanguageToHex( const s : string) : string');
18008:   Function MSHexToLanguage( const s : string) : string');
18009:   Function MSCharacterSetToHex( const s : string) : string');
18010:   Function MSHexToCharacterSet( const s : string) : string');
18011:   Function MSLanguages : TStringList');
18012:   Function MSHexLanguages : TStringList');
18013:   Function MSCharacterSets : TStringList');
18014:   Function MSHexCharacterSets : TStringList');
18015: end;
18016:
18017: procedure SIRегистre_IpUtils(CL: TPPascalCompiler);
18018: begin
18019:   TIpHandle', 'Cardinal');
18020:   TIpMD5StateArray', 'array[0..3] of DWORD;');
18021:   CL.AddTypeS('TIpMD5CountArray', 'array[0..1] of DWORD;');
18022:   TIpMD5ByteBuf', 'array[0..63] of Byte;');
18023:   TIpMD5LongBuf', 'array[0..15] of DWORD;');
18024:   TIpMD5Digest', 'array[0..15] of Byte;');
18025:   TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther )');
18026:   TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5';
18027:     +CountArray; ByteBuf : TIpMD5ByteBuf; end');
18028:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpBaseException');
18029:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpAccessException');
18030:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EIpHtmlException');
18031:   SIRегистre_TIpBaseAccess(CL);
18032:   SIRегистre_TIpBasePersistent(CL);
18033:   //TIPComponentClass', 'class of TIpBaseComponent';
18034:   SIRегистre_TIpBaseComponent(CL);
18035:   SIRегистre_TIpBaseWinControl(CL);
18036:   Function InClassA( Addr : LongInt) : Boolean');
18037:   Function InClassB( Addr : LongInt) : Boolean');
18038:   Function InClassC( Addr : LongInt) : Boolean');
18039:   Function InClassD( Addr : LongInt) : Boolean');
18040:   Function InMulticast( Addr : LongInt) : Boolean');
18041:   Function IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar) : DWORD');
18042:   Function IpCompStruct( const S1, S2, Size : Cardinal) : Integer');
18043:   Function IpMaxInt( A, B : Integer) : Integer');
18044:   Function IpMinInt( A, B : Integer) : Integer');
18045:   Procedure IpSafeFree( var Obj: TObject);
18046:   Function IpShortVersion : string');
18047:   Function IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD) : DWORD');
18048:   Function IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD) : DWORD');
18049:   Function IpInternetSumOfFile( const FileName : string) : DWORD');
18050:   Function MD5SumOfFile( const FileName : string) : string');
18051:   Function MD5SumOfStream( Stream : TStream) : string');
18052:   Function MD5SumOfStreamDigest( Stream : TStream) : TIpMD5Digest');
18053:   Function MD5SumOfString( const S : string) : string');
18054:   Function MD5SumOfStringDigest( const S : string) : TIpMD5Digest');
18055:   Function SafeYield : LongInt';
18056:   Function AllTrimSpaces( Strng : string) : string');

```

```

18057: Function IpCharPos( C : AnsiChar; const S : string) : Integer';
18058: Function CharPosIdx( C : AnsiChar; const S : string; Idx : Integer) : Integer';
18059: Function NthCharPos( C : AnsiChar; const S : string; Nth : Integer) : Integer';
18060: Function RCharPos( C : AnsiChar; const S : string) : Integer';
18061: Function RCharPosIdx( C : AnsiChar; const S : string; Idx : Integer) : Integer';
18062: Function RNthCharPos( C : AnsiChar; const S : string; Nth : Integer) : Integer';
18063: Function IpRPos( const Substr : string; const S : string) : Integer';
18064: Function IpPosIdx( const SubStr, S : string; Idx : Integer) : Integer';
18065: ACharSet', 'set of AnsiChar');
18066: TIpAddrRec', 'record Scheme : string; UserName : string; Password string; Authority : string;
18067: Port : string; Path:string; Fragment: string; Query : string; QueryDelim : AnsiChar; end');
18068: Procedure Initialize( var AddrRec : TIpAddrRec)');
18069: Procedure Finalize( var AddrRec : TIpAddrRec)');
18070: Function ExtractEntityName( const NamePath : string) : string');
18071: Function ExtractEntityPath( const NamePath : string) : string');
18072: Function IpParseURL( const URL : string; var Rslt : TIpAddrRec) : Boolean');
18073: Function BuildURL( const OldURL, NewURL : string) : string');
18074: Function PutEscapes( const S : string; EscapeSet : ACharSet) : string');
18075: Function RemoveEscapes( const S : string; EscapeSet : ACharSet) : string');
18076: Procedure SplitParams( const Params : string; Dest : TStrings)');
18077: Function NetToDOSPath( const PathStr : string) : string');
18078: Function DOSToNetPath( const PathStr : string) : string');
18079: Procedure SplitHttpResponse( const S : string; var V, MsgID, Msg : string)');
18080: Procedure FieldFix( Fields : TStrings)');
18081: Function AppendSlash( APath : string) : string');
18082: Function RemoveSlash( APath : string) : string');
18083: Function GetParentPath( const Path : string) : string');
18084: Function GetLocalContent( const TheFileName : string) : string');
18085: Function IPDirExists( Dir : string) : Boolean');
18086: Function GetTemporaryFile( const Path : string) : string');
18087: Function GetTemporaryPath : string');
18088: Function AppendBackSlash( APath : string) : string');
18089: Function IpRemoveBackSlash( APath : string) : string');
18090: Function INetDateStrToDate( const DateStr : string) : TDateTime');
18091: Function DateToINetDateStr( DateTime : TDateTime) : string');
18092: Function IpTimeZoneBias : Integer');
18093: Procedure SplitCookieFields( const Data : string; Fields : TStrings)');
18094: end;
18095:
18096: procedure SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18097: begin
18098:   CL.AddTypeS('TPOStyle', '( postStandard, postPropName, postFull )');
18099:   Procedure Lrt2Po( const LRTFile : string; PostStyle : TPOStyle)');
18100:   Procedure CombinePoFiles( SL : TStrings; const FName : string)');
18101: end;
18102:
18103: procedure SIRegister_GPS(CL: TPSPascalCompiler);
18104: begin
18105:   CL.AddConstantN('MAX_SATS','LongInt').SetInt( 12 );
18106:   GPSMSG_START', 'String').SetString( '$' );
18107:   GPSMSG_STOP', 'String').SetString( '*' );
18108:   SEC_BETWEEN_SEG', 'LongInt').SetInt( 5 );
18109:   CL.AddTypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor'
18110:     +tint; Azimut : Smallint; SignLevel : Smallint; end');
18111:   CL.AddTypeS('TSatellites', 'array[1..12] of TSatellite');
18112:   //TSatellites = array[1..MAX_SATS] of TSatellite;
18113:   TGPSSatEvent', 'Procedure(Sender: TObject; NbSat, NbSatUse: Shortint; Sats : TSatellites)');
18114:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Heigh'
18115:     tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid : Boolean; '
18116:     NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end');
18117:   CL.AddTypeS('TGPSDatasEvent', 'Procedure ( Sender : TObject; GPSDatas : TGPSDatas)');
18118:   CL.AddTypeS('TMsgGP', '( msgGP,msgGPGGA,msgGPGLL,msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA )');
18119:   CL.AddTypeS('TSpeedUnit', '( suKilometre, suMile, suNauticalMile )');
18120:   SIRegister_TGPSLink(CL);
18121:   SIRegister_TCustomGPS(CL);
18122:   SIRegister_TGPS(CL);
18123:   SIRegister_TGPSStoGPX(CL);
18124:   SIRegister_TGPSSpeed(CL);
18125:   SIRegister_TGPSSatellitesPosition(CL);
18126:   SIRegister_TGPSSatellitesReception(CL);
18127:   SIRegister_TGPSCompass(CL);
18128:   //Procedure Register( );
18129:   Function IndexMsgGP( StrMsgGP : String) : TMsgGP');
18130:   Function StrCoordToAngle( Point : Char; Angle : String) : Double');
18131:   Function StrTimeToTime( const Time : String) : TDateTime');
18132:   Function StrToInteger( const Str : String) : Integer');
18133:   Function StrToReal( const Str : String) : Extended');
18134:   Function GPSRotatePoint( Angle : Double; Ct, Pt : TPoint) : TPoint');
18135:   Procedure LoadResource( RessourceName : String; ImageList : TImageList)');
18136: end;
18137:
18138: procedure SIRegister_NMEA(CL: TPSPascalCompiler);
18139: begin
18140:   NMEADataArray', 'array of string');
18141:   Procedure TrimNMEA( var S : string)');
18142:   Procedure ExpandNMEA( var S : string)');
18143:   Function ParseNMEA( S : string) : NMEADataArray');
18144:   Function ChkValidNMEA( S : string) : Boolean');
18145:   Function IdNMEA( S : string) : string');

```

```

18146: Function ChkSumNMEA( const S : string ) : string');
18147: Function PosInDeg( const PosStr : string ) : Double');
18148: Function DateTimeNMEA( const StrD, StrT : string ) : TDateTime');
18149: Function SysClockSet( const StrD, StrT : string ) : Boolean');
18150: function Ticks2Secs(Ticks : LongInt) : LongInt;');
18151: function Secs2Ticks(Secs : LongInt) : LongInt;');
18152: function MSecs2Ticks(MSecs : LongInt) : LongInt;');
18153: end;
18154:
18155: procedure SIRegister_SortUtils(CL: TPSPascalCompiler);
18156: begin
18157:   CL.AddTypeS('SortType1', 'Byte');
18158:   CL.AddTypeS('SortType2', 'Double');
18159:   CL.AddTypeS('SortType3', 'DWord');
18160:   //CL.AddTypeS('PDWordArray', '^DWordArray // will not work');
18161:   CL.AddTypeS('TDataRecord4', 'record Value : Integer; Data : Integer; end');
18162: Function('Procedure QuickSort( var List : array of SortType1; Min, Max : Integer)');
18163: Procedure QuickSortDWord( var List : array of SortType3; Min, Max : Integer)');
18164: Procedure QuickSortDataRecord4( var List : array of TDataRecord4; Count : Integer)');
18165: Procedure HeapSort( var List : array of SortType1; Count : DWord; FirstNeeded : DWord)');
18166: Function QuickSelect(var List : array of SortType1; Min, Max, Wanted : Integer) : SortType1');
18167: Function QuickSelectDouble(var List : array of SortType2; Min, Max, Wanted : Integer) : SortType2');
18168: Function QuickSelectDWord(var List : array of SortType3; Min, Max, Wanted : Integer) : SortType3');
18169: end;
18170:
18171: procedure SIRegister_BitmapConversion(CL: TPSPascalCompiler);
18172: begin
18173:   // TMatrix3x3 = array[1..3,1..3] of Double;
18174:   // TMatrix4x4 = array[1..4,1..4] of Double;
18175:   CL.AddTypeS('TMatrix3x31', 'array[1..3] of Double');
18176:   CL.AddTypeS('TMatrix3x33', 'array[1..3] of TMatrix3x31');
18177:   CL.AddTypeS('TMatrix4x41', 'array[1..4] of Double');
18178:   CL.AddTypeS('TMatrix4x44', 'array[1..4] of TMatrix4x41');
18179:   Procedure ColorTransform( A, B, C : Byte; out X, Y, Z : Byte; const T : TMatrix4x4 );
18180:   Procedure ColorTransform1( A, B, C : Byte; out X, Y, Z : Float; const T : TMatrix4x4 );
18181:   Procedure ColorTransform2(const A,B,C : Float; out X, Y, Z : Byte; const T : TMatrix4x4 );
18182:   Procedure ColorTransformHSI2RGB( H, S, I : Byte; out R, G, B : Byte );
18183:   Procedure ColorTransformRGB2HSI( R, G, B : Byte; out H, S, I : Byte );
18184:   Procedure ColorTransformRGB2Lab( R, G, B : Byte; out L, a_, b_ : Byte );
18185:   Procedure ColorTransformLab2RGB( L, a_, b_ : Byte; out R, G, B : Byte );
18186:   Procedure ColorTransformRGB2LOCO( R, G, B : Byte; out S0, S1, S2 : Byte );
18187:   Procedure ColorTransformLOCO2RGB( S0, S1, S2 : Byte; out R, G, B : Byte );
18188:   Procedure ConvertColorSpace(Image:TLinearBitmap; const T:TMatrix4x4; NewImage: TLinearBitmap );
18189:   //Procedure
18190:   ConvertColorSpace1(Image:TLinearBitmap;ColorTransform:TColorTransformProc;NewImage:TLinearBitmap );
18191:   Procedure ConvertToGrayscale( const Image, GrayImage : TLinearBitmap );
18192:   Procedure ConvertToGrayscale1( const Image : TLinearBitmap );
18193:
18194: procedure SIRegister_ZDbcUtils(CL: TPSPascalCompiler);
18195: begin
18196:   { TZSQLType = (zsqlstUnknown, zsqlstBoolean, zsqlstByte, zsqlstShort, zsqlstInteger, zsqlstLong,
18197:     zsqlstFloat,zsqlstDouble,zsqlstBigDecimal, zsqlstString, zsqlstUnicodeString, zsqlstBytes,
18198:     zsqlstDate, zsqlstTime, zsqlstTimestamp, zsqlstDataSet, zsqlstGUID,
18199:     stAsciiStream, stUnicodeStream, stBinaryStream);}
18200:   Function ResolveConnectionProtocol(Url:string; SupportedProtocols:TStringDynArray) : string';
18201:   //Procedure ResolveDatabaseUrl( const Url: string; Info:TStrings; var HostName:string; var
18202:   Port:Integer;var Database : string; var UserName : string; var Password : string; ResultInfo : TStrings );
18203:   Function CheckConversion( InitialType : TZSQLType; ResultType : TZSQLType ) : Boolean';
18204:   Procedure DefineColumnType( ColumnType : TZSQLType ) : string';
18205:   Procedure RaiseSQLException( E : Exception );
18206:   //Procedure CopyColumnInfo( FromList : TObjectList; ToList : TObjectList );
18207:   Function ToLikeString( const Value : string ) : string';
18208:   Function GetSQLHexWideString( Value : PChar; Len : Integer; ODBC : Boolean ) : WideString';
18209:   Function GetSQLHexAnsiString( Value : PChar; Len : Integer; ODBC : Boolean ) : RawByteString';
18210:   Function WideStringStream( const AString : WideString ) : TStream';
18211:   function ConvertAdoToTypeName(FieldType: SmallInt) : string';
18212:   function GetTableName(const AField: TField) : string';
18213:   function GetFieldName(const AField: TField) : string';
18214: end;
18215:
18216: procedure SIRegister_JclTD32(CL: TPSPascalCompiler);
18217:   CL.AddTypeS('TSymbolInfo', 'record Size : Word; SymbolType : Word; end');
18218:   //CL.AddTypeS('PSymbolInfos', '^TSymbolInfos // will not work');
18219:   SIRegister_TJclModuleInfo(CL);
18220:   SIRegister_TJclLineInfo(CL);
18221:   SIRegister_TJclSourceModuleInfo(CL);
18222:   SIRegister_TJclSymbolInfo(CL);
18223:   SIRegister_TJclProcSymbolInfo(CL);
18224:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclLocalProcSymbolInfo');
18225:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TJclGlobalProcSymbolInfo');
18226:   SIRegister_TJclTD32InfoParser(CL);
18227:   SIRegister_TJclTD32InfoScanner(CL);
18228:   SIRegister_TJclPeBorTD32Image(CL);
18229: end;
18230:
18231: procedure SIRegister_JvIni(CL: TPSPascalCompiler);
18232: begin

```

```

18233: CL.AddTypeS('TReadObjectEvent', 'Function ( Sender : TObject; const Section, '
18234: +'Item, Value : string) : TObject');
18235: TWriteObjectEvent', 'Procedure(Sender: TObject; const Section, Item: string; Obj: TObject)');
18236: Function StringToFontStyles( const Styles : string) : TFontStyles');
18237: Function FontsToStrings( Styles : TFontStyles) : string');
18238: Function FontToString( Font : TFont) : string');
18239: Procedure StringToFont( const Str : string; Font : TFont)');
18240: Function RectToStr( Rect : TRect) : string');
18241: Function StrToRect( const Str : string; const Def : TRect) : TRect');
18242: Function JPointToStr( P : TPoint) : string');
18243: Function JStrToPoint( const Str : string; const Def : TPoint) : TPoint');
18244: Function DefProfileName : string');
18245: Function DefLocalProfileName : string');
18246: CL.AddConstantN('idnListItem','String').SetString( 'Item');
18247: end;
18248:
18249: procedure SIRegister_JvHtControls(CL: TPSPascalCompiler);
18250: begin
18251: Procedure ItemHtDrawEx( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string; const HideSelColor:Boolean; var PlainItem: string; var Width: Integer; CalcWidth:Boolean)');
18252: Function ItemHtDraw( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string; const HideSelColor : Boolean) : string');
18253: Function ItemHtWidth( Canvas : TCanvas; Rect : TRect; const State : TOwnerDrawState; const Text : string; const HideSelColor : Boolean) : Integer');
18254: Function ItemHtPlain( const Text : string) : string');
18255: Procedure ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:str);
18256: Function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string;MouseX,MouseY:Integer;var HyperLink:string):Bool;
18257: end;
18258:
18259: procedure SIRegister_NeuralNetwork(CL: TPSPascalCompiler);
18260: begin
18261: CL.AddClassN(CL.FindClass('TOBJECT'),'TNeuron');
18262: CL.AddTypeS('TSynapse', 'record W : Real; Connection : TNeuron; end');
18263: CL.AddTypeS('TAcson', 'record Alfa : Real; Beta : Real; Gama : Real; end');
18264: SIRegister_TNeuron(CL);
18265: SIRegister_TNeuronLayer(CL);
18266: SIRegister_TNeuralNet(CL);
18267: end;
18268:
18269: procedure SIRegister_StExpr(CL: TPSPascalCompiler);
18270: begin
18271: //CL.AddTypeS('PStFloat', '^TStFloat // will not work');
18272: TStMethod0Param', 'Function : TStFloat');
18273: TStMethod1Param', 'Function ( Value1 : TStFloat) : TStFloat');
18274: TStMethod2Param', 'Function ( Value1, Value2 : TStFloat) : TStFloat');
18275: TStMethod3Param', 'Function ( Value1, Value2, Value3 : TStFloat) : TStFloat');
18276: TStGetIdentValueEvent', 'Procedure ( Sender : TObject; const Ide'
18277: +'ntifier : AnsiString; var Value : TStFloat)');
18278: TStToken', '( ssStart, ssInIdent, ssInNum, ssInSign, ssInExp, ss'
18279: +'Eol,ssNum,ssIdent,ssLPar,ssRPar,ssComma,ssPlus,ssMinus,ssTimes,ssDiv,ssEqual,ssPower )');
18280: SIRegister_TStExpression(CL);
18281: TStExprErrorEvent', 'Procedure ( Sender : TObject; ErrorNumber :'
18282: +' LongInt; const ErrorStr : AnsiString)');
18283: SIRegister_TStExpressionEdit(CL);
18284: Function AnalyzeExpr( const Expr : AnsiString) : Double';
18285: Procedure TpVal( const S : AnsiString; var V : Extended; var Code : Integer)');
18286: end;
18287:
18288: procedure SIRegister_GR32_Containers(CL: TPSPascalCompiler);
18289: begin
18290: CL.AddConstantN('BUCKET_MASK','LongWord').SetUInt( $FF);
18291: CL.AddConstantN('BUCKET_COUNT','LongInt').SetInt( BUCKET_MASK + 1);
18292: Procedure SmartAssign( Src, Dst : TPersistent; TypeKinds : TTypeKinds)');
18293: Procedure Advance( var Node : TLinkedNode; Steps : Integer)');
18294: end;
18295:
18296: procedure SIRegister_StSaturn(CL: TPSPascalCompiler);
18297: begin
18298: TStUpSatPos', 'record X: double; Y: Double; end');
18299: TStJupSats record Io:TStJupSatPos;Europa:TStJupSatPos;Ganymede:TStJupSatPos;Callisto:TStJupSatPos;end;
18300: Function ComputeSaturn( JD : Double) : TStEclipticalCord');
18301: Function ComputePluto( JD : Double) : TStEclipticalCord');
18302: Function ComputeVenus( JD : Double) : TStEclipticalCord');
18303: Function ComputeMars( JD : Double) : TStEclipticalCord');
18304: Function ComputeMercury( JD : Double) : TStEclipticalCord');
18305: Function ComputeJupiter( JD : Double) : TStEclipticalCord');
18306: Function ComputeUranus( JD : Double) : TStEclipticalCord');
18307: Function ComputeNeptune( JD : Double) : TStEclipticalCord');
18308: function GetJupSats(JD : TDateTime; HighPrecision, Shadows : Boolean) : TStJupSats;');
18309: end;
18310:
18311: procedure SIRegister_JclParseUses(CL: TPSPascalCompiler);
18312: begin
18313: CL.AddClassN(CL.FindClass('TOBJECT'),'EUsesListError');
18314: Function CreateGoal( Text : PChar) : TCustomGoal');
18315: end;
18316:
18317: procedure SIRegister_JvFinalize(CL: TPSPascalCompiler);

```

```

18318: begin
18319: //type
18320: //  TFinalizeProc = procedure;
18321:   CL.AddTypeS('TFinalizeProc', 'procedure');
18322: Procedure AddFinalizeProc( const UnitName : string; FinalizeProc : TFinalizeProc );
18323: Function AddFinalizeObject( const UnitName : string; Instance : TObject ) : TObject';
18324: Function AddFinalizeObjectNil( const UnitName : string; var Reference: TObject ) : TObject';
18325: Function AddFinalizeFreeAndNil( const UnitName : string; var Reference: TObject ) : TObject';
18326: Function AddFinalizeMemory( const UnitName : string; Ptr : __Pointer ) : __Pointer';
18327: Function AddFinalizeMemoryNil( const UnitName : string; var Ptr: __Pointer ) : __Pointer';
18328: Procedure FinalizeUnit( const UnitName : string )';
18329: end;
18330:
18331: procedure SIRегистер_BigIni(CL: TPPascalCompiler);
18332: begin
18333:   CL.AddConstantN('IniTextBufferSize','LongWord').SetUInt( $7000 );
18334:   CL.AddConstantN('cIniCount','String').SetString( 'Count' );
18335:   TEraseSectionCallback', 'Function ( const sectionName : string; '
18336:     const s11, s12 : TStringList ) : Boolean';
18337:   SIRегистер_TCommaSeparatedInfo(CL);
18338:   SIRегистер_TSectionList(CL);
18339:   SIRегистер_TBigIniFile(CL);
18340:   SIRегистер_TBiggerIniFile(CL);
18341:   SIRегистер_TAppIniFile(CL);
18342:   SIRегистер_TLibIniFile(CL);
18343:   Function ModuleName( getLibraryName : Boolean ) : String';
18344: end;
18345:
18346: procedure SIRегистер_ShellCtrls(CL: TPPascalCompiler);
18347: begin
18348:   CL.AddTypeS('TRoot', 'string');
18349:   CL.AddTypeS('TRootFolder', '( rfDesktop, rfMyComputer, rfNetwork, rfRecycleBi'
18350:     +'n, rfAppData, rfCommonDesktopDirectory, rfCommonPrograms, rfCommonStartMen'
18351:     +'u, rfCommonStartup, rfControlPanel, rfDesktopDirectory, rfFavorites, rfFon'
18352:     +'ts, rfInternet, rfPersonal, rfPrinters, rfPrintHood, rfPrograms, rfRecent, '
18353:     +'rfSendTo, rfStartMenu, rfStartup, rfTemplates )');
18354:   TShellFolderCapability', '( fcCanCopy, fcCanDelete, fcCanLink, f'
18355:     +'cCanMove, fcCanRename, fcDropTarget, fcHasPropSheet )');
18356:   TShellFolderCapabilities', 'set of TShellFolderCapability';
18357:   TShellFolderProperty', '( fpCut, fpIsLink, fpReadOnly, fpShared, '
18358:     +'fpFileSystem, fpFileSystemAncestor, fpRemovable, fpValidate )';
18359:   TShellFolderProperties', 'set of TShellFolderProperty)';
18360:   TShellObjectType', '( otFolders, otNonFolders, otHidden )');
18361:   TShellObjectTypes', 'set of TShellObjectType)';
18362:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EInvalidPath');
18363:   SIRегистер_IShellCommandVerb(CL);
18364:   SIRегистер_TShellFolder(CL);
18365:   TNotifyFilter', '( nfFileNameChange, nfDirNameChange, nfAttribut'
18366:     +'eChange, nfSizeChange, nfWriteChange, nfSecurityChange )');
18367:   TNotifyFilters', 'set of TNotifyFilter)';
18368:   SIRегистер_TShellChangeThread(CL);
18369:   SIRегистер_TCustomShellChangeNotifier(CL);
18370:   SIRегистер_TShellChangeNotifier(CL);
18371:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TCustomShellComboBox');
18372:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TCustomShellListView');
18373:   TAddFolderEvent', 'Procedure ( Sender : TObject; AFolder : TShel'
18374:     +'lFolder; var CanAdd : Boolean )';
18375:   TGetImageIndexEvent', 'Procedure ( Sender : TObject; Index : Int'
18376:     +'eger; var ImageIndex : Integer )';
18377:   SIRегистер_TCustomShellTreeView(CL);
18378:   SIRегистер_TShellTreeView(CL);
18379:   SIRегистер_TCustomShellComboBox(CL);
18380:   SIRегистер_TShellComboBox(CL);
18381:   SIRегистер_TCustomShellListView(CL);
18382:   SIRегистер_TShellListView(CL);
18383:   Procedure InvokeContextMenu(Owner : TWinControl; AFolder : TShellFolder; X, Y : Integer)' );
18384: end;
18385:
18386: procedure SIRегистер_fmath(CL: TPPascalCompiler);
18387: begin
18388:   CL.AddTypeS('Float', 'Double');
18389:   'FN_OK','LongInt').SetInt( 0 );
18390:   CL.AddConstantN('FN_DOMAIN','LongInt').SetInt( - 1 );
18391:   'FN_SING','LongInt').SetInt( - 2 );
18392:   'FN_OVERFLOW','LongInt').SetInt( - 3 );
18393:   'FN_UNDERFLOW','LongInt').SetInt( - 4 );
18394:   'FN_TLOSS','LongInt').SetInt( - 5 );
18395:   'FN_PLOSS','LongInt').SetInt( - 6 );
18396:   //CL.AddConstantN('NFACT','LongInt').SetInt( 33 );
18397:   Function MathError : Integer';
18398:   Function FMin( X, Y : Float ) : Float';
18399:   Function FMax( X, Y : Float ) : Float';
18400:   Function IMin( X, Y : Integer ) : Integer';
18401:   Function IMax( X, Y : Integer ) : Integer';
18402:   Function FSgn( X : Float ) : Integer';
18403:   Function Sgn0( X : Float ) : Integer';
18404:   Function DSgn( A, B : Float ) : Float';
18405:   Procedure FSwap( var X, Y : Float );
18406:   Procedure ISwap( var X, Y : Integer );

```

```

18407: Function fExpo( X : Float ) : Float';
18408: Function fExp2( X : Float ) : Float';
18409: Function fExp10( X : Float ) : Float';
18410: Function fLog( X : Float ) : Float';
18411: Function fLog2( X : Float ) : Float';
18412: Function fLog10( X : Float ) : Float';
18413: Function fLogA( X, A : Float ) : Float';
18414: Function fIntPower( X : Float; N : Integer ) : Float';
18415: Function fPower( X, Y : Float ) : Float';
18416: Function Pythag( X, Y : Float ) : Float';
18417: Function FixAngle( Theta : Float ) : Float';
18418: Function fTan( X : Float ) : Float';
18419: Function fArcSin( X : Float ) : Float';
18420: Function fArcCos( X : Float ) : Float';
18421: Function fArcTan2( Y, X : Float ) : Float';
18422: Procedure fSinCos( X : Float; var SinX, CosX : Float );
18423: Function fSinh( X : Float ) : Float';
18424: Function fCosh( X : Float ) : Float';
18425: Function fTanh( X : Float ) : Float';
18426: Function fArcSinh( X : Float ) : Float';
18427: Function fArcCosh( X : Float ) : Float';
18428: Function fArcTanh( X : Float ) : Float';
18429: Procedure fSinhCosh( X : Float; var SinhX, CoshX : Float );
18430: Function fFact( N : Integer ) : Float';
18431: Function fBinomial( N, K : Integer ) : Float';
18432: Function fGamma( X : Float ) : Float';
18433: Function fSgnGamma( X : Float ) : Integer';
18434: Function LnGamma( X : Float ) : Float';
18435: Function fIGamma( A, X : Float ) : Float';
18436: Function fJGamma( A, X : Float ) : Float';
18437: Function fBeta( X, Y : Float ) : Float';
18438: Function fIBeta( A, B, X : Float ) : Float';
18439: Function fErf( X : Float ) : Float';
18440: Function fErfc( X : Float ) : Float';
18441: Function fPBinom( N: Integer; P: Float; K : Integer ) : Float';
18442: Function FBinom( N: Integer; P: Float; K : Integer ) : Float';
18443: Function PPoisson( Mu : Float; K : Integer ) : Float';
18444: Function FPoisson( Mu : Float; K : Integer ) : Float';
18445: Function fDNorm( X : Float ) : Float';
18446: Function FNorm( X : Float ) : Float';
18447: Function PNorm( X : Float ) : Float';
18448: Function InvNorm( P : Float ) : Float';
18449: Function fDStudent( Nu : Integer; X : Float ) : Float';
18450: Function FStudent( Nu : Integer; X : Float ) : Float';
18451: Function PStudent( Nu : Integer; X : Float ) : Float';
18452: Function fDKhi2( Nu : Integer; X : Float ) : Float';
18453: Function FKhi2( Nu : Integer; X : Float ) : Float';
18454: Function PKhi2( Nu : Integer; X : Float ) : Float';
18455: Function fDSnedecor( Nu1, Nu2 : Integer; X : Float ) : Float';
18456: Function FSnedecor( Nu1, Nu2 : Integer; X : Float ) : Float';
18457: Function PSnedecor( Nu1, Nu2 : Integer; X : Float ) : Float';
18458: Function fDExpo( A, X : Float ) : Float';
18459: Function FEExpo( A, X : Float ) : Float';
18460: Function fDBeta( A, B, X : Float ) : Float';
18461: Function FBeta( A, B, X : Float ) : Float';
18462: Function fDGamma( A, B, X : Float ) : Float';
18463: Function FGamma( A, B, X : Float ) : Float';
18464: Procedure RMarIn( Seed1, Seed2 : Integer );
18465: Function IRanMar : LongInt';
18466: Function RanMar : Float';
18467: Function RanGaussStd : Float';
18468: Function RanGauss( Mu, Sigma : Float ) : Float';
18469: end;
18470:
18471: procedure SIRegister_fcomp(CL: TPSPascalCompiler);
18472: begin
18473:   CL.AddTypeS('ComplexForm', '( Rec, Pol )');
18474:   CL.AddTypeS('TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R : '
18475:     + 'Float; Theta : Float; end');
18476:   Procedure CSet( var Z : TComplex; A, B : Float; F : ComplexForm );
18477:   Procedure CConvert( var Z : TComplex; F : ComplexForm );
18478:   Procedure CSwap( var X, Y : TComplex );
18479:   Function CReal( Z : TComplex ) : Float';
18480:   Function CImag( Z : TComplex ) : Float';
18481:   Function CAbs( Z : TComplex ) : Float';
18482:   Function CArg( Z : TComplex ) : Float';
18483:   Function CSgn( Z : TComplex ) : Integer';
18484:   Procedure CNeg( A : TComplex; var Z : TComplex );
18485:   Procedure CConj( A : TComplex; var Z : TComplex );
18486:   Procedure CAdd( A, B : TComplex; var Z : TComplex );
18487:   Procedure CSub( A, B : TComplex; var Z : TComplex );
18488:   Procedure CDiv( A, B : TComplex; var Z : TComplex );
18489:   Procedure CMult( A, B : TComplex; var Z : TComplex );
18490:   Procedure CLn( A : TComplex; var Z : TComplex );
18491:   Procedure CExp( A : TComplex; var Z : TComplex );
18492:   Procedure CPower( A, B : TComplex; var Z : TComplex );
18493:   Procedure CIntPower( A : TComplex; N : Integer; var Z : TComplex );
18494:   Procedure CRealPower( A : TComplex; X : Float; var Z : TComplex );
18495:   Procedure CSqrt( A : TComplex; var Z : TComplex );

```

```

18496: Procedure CRoot( A : TComplex; K, N : Integer; var Z : TComplex)'';
18497: Procedure CCSin( A : TComplex; var Z : TComplex)'';
18498: Procedure CCos( A : TComplex; var Z : TComplex)'';
18499: Procedure CTan( A : TComplex; var Z : TComplex)'';
18500: Procedure CArcSin( A : TComplex; var Z : TComplex)'';
18501: Procedure CArcCos( A : TComplex; var Z : TComplex)'';
18502: Procedure CArcTan( A : TComplex; var Z : TComplex)'';
18503: Procedure CSinh( A : TComplex; var Z : TComplex)'';
18504: Procedure CCosh( A : TComplex; var Z : TComplex)'';
18505: Procedure CTanh( A : TComplex; var Z : TComplex)'';
18506: Procedure CArcSinh( A : TComplex; var Z : TComplex)'';
18507: Procedure CArcCosh( A : TComplex; var Z : TComplex)'';
18508: Procedure CArcTanh( A : TComplex; var Z : TComplex)'';
18509: Procedure CLnGamma( A : TComplex; var Z : TComplex)'';
18510: end;
18511;
18512: procedure SIRegister_XSBuiltIns(CL: TPPascalCompiler);
18513: begin
18514:   CL.AddConstantN('SHexMarker','String').SetString( '$');
18515:   Function DateTimeToXMLTime( Value : TDateTime; ApplyLocalBias : Boolean) : WideString';
18516:   Function XMLTimeToDate( const XMLDate : WideString; AsUTC : Boolean) : TDateTime';
18517:   Function DateTimeToXSDDate( const Value: TDateTime; ApplyLocalBias: Boolean): TXSDDate';
18518:   Function GetDataFromFile( AFileName : string) : string';
18519:   Function SoapFloatToStr( Value : double) : string';
18520:   Function SoapStrToFloat( Value : string) : double';
18521:   Procedure InitXSTypes';
18522: end;
18523;
18524: procedure SIRegister_CompFileIo(CL: TPPascalCompiler);
18525: begin
18526:   SIRegister_TComponentStream(CL);
18527:   CL.AddTypeS('TComponentStream', 'TMemoryStream');
18528:   CL.AddTypeS('TComponentFormat', '( cffText, cffBinary )');
18529:   CL.AddTypeS('TComponentArray', 'array of TComponent');
18530:   TResourceNaming', 'rnClassNameTag,rnClassName,rnClassTag,rnNameTag,rnClass,rnName, rnTag ');
18531:   Function VOID_COMP : __Pointer';
18532:   Function VOID_FORM : __Pointer';
18533:   Function CreateIoForm( AForm : TCustomForm; ClassType : TFormClass) : TCustomForm';
18534:   Function GetComponentTree(Component:TComponent;pComponents:TComponentArray;bAddSelSelf:Bool):LongInt;
18535:   Function pPosEx( SearchStr: PChar; Str : PChar; var Pos : LongInt) : PChar';
18536:   Function pGetTextBetween(pBuff:PChar;bSearchCode:string;eSearchCode:string;Container:TStrings) : LongInt;
18537:   Function pComponentToString( Component : TComponent) : string';
18538: //Function StringToComponent(Value:string; ComponentClass : TComponentClass) : TComponent';
18539:   Function StringToObjectBinaryStream(Value: string;BinStream:TMemoryStream;ResName:string) : Boolean;
18540:   Function ObjectBinaryStreamToString(BinStream:TMemoryStream;var sResult:string;
bResource:Boolean);
18541:   Function ObjectTextToBinaryStream(StrStream,BinStream:TComponentStream;ObjCount:LongInt) : LongInt;
18542:   Function ObjectBinaryStreamToObjectTextStream( BinStream : TMemoryStream; StrStream : TMemoryStream;
bResource : Boolean) : Boolean';
18543:   Function GetResHeaderInfo( Stream : TMemoryStream; sl : TStrings) : Boolean';
18544:   Function pGetResourceName(Component: TComponent; NamingMethod : TResourceNaming) : string';
18545:   Function GetFormResourceStream(Form:TCustomForm;ResName:string;var ResourceStream:TMemoryStream):Bool;
18546:   Function WriteComponentsToFile( FormsAndComponents : array of TComponent; FileName : string; Format : TComponentFileFormat; StoreComponentNames : Boolean) : Boolean';
18547:   Function ReadComponentsFromFile(FormsAndComponents: array of TComponent;FileName:string):Bool;
18548: //Function ReadComponentsFromFile( FileName : string; pComponents : array of TPComponent;
ComponentClasses : array of TComponentClass) : Boolean';
18549:   Function ReadComponentTreeFromFile(FormOrComponent:TComponent; FileName : string):Boolean';
18550:   Function WriteComponentToFile( Component : TComponent; FileName : string; Format : TComponentFileFormat;
StoreComponentName : Boolean) : Boolean';
18551:   Function WriteComponentTreeToFile( FormOrComponent : TComponent; FileName : string; Format :
TComponentFileFormat; StoreComponentName : Boolean) : Boolean';
18552:   Function ReadComponentFromFile4(FormOrComponent:TComponent;FileName : string): Boolean;';
18553:   Function ReadFormFromFile(pInstance: TComponent; FormClass:TFormClass; FileName: string) : Boolean';
18554:   Function ReadComponentResourceFile5( Instance : TObject; FileName : string) : Boolean';
18555:   Function WriteComponentResourceFile(instance:TObject;FileName:string;StoreFormAsVisible:Boolean):Boolean;
18556:   Function ReadComponentsResourceFile( Components : array of TComponent; FileName : string; NamingMethod : TResourceNaming; bLoadTotalForm : Boolean) : Boolean';
18557:   Function WriteComponentsResourceFile( Components : array of TComponent; FileName : string; NamingMethod : TResourceNaming) : Boolean';
18558:   Function ReadComponentResourceHeader( sHeaderInfo : TStrings; pSize : Integer; FileName : string;
NamingMethod : TResourceNaming) : Boolean';
18559:   Function ConvertComponentResourceToTextFile(SourceFileName:string; TargetFileName : string;
bStoreResNames : Boolean) : Boolean';
18560:   Function
CheckComponentInResourceFile(Component:TComponent;FileName:string;NamingMethod:TResourceNaming):Boolean;
18561:   Function DeleteComponentFromResourceFile(ResourceName:string; FileName:string) : Boolean;';
18562:   Function DeleteComponentFromResourceFile8( Component : TComponent; FileName : string; NamingMethod : TResourceNaming) : Boolean';
18563: //Function CreateFormFromResFile( AOwner : TComponent; NewClassType : TFormClass; FileName : string) :
Pointer';
18564: //Function CreateComponentFromResFile( AOwner : TComponent; NewClassType : TComponentClass; FileName :
string) : Pointer';
18565: end;
18566;
18567: procedure SIRegister_SMScript(CL: TPPascalCompiler);
18568: begin
18569:   CL.FindClass('TOBJECT','TSMSScriptExecutor');
18570:   SIRegister_TSMSError(CL);

```

```

18571: CL.AddClassN(CL.FindClass('TOBJECT'), 'TSMSEModule');
18572: CL.AddTypeS('TSMSEProcedureType', '( ptProcedure, ptFunction )');
18573: SIRegister_TSMSEProcedure(CL);
18574: SIRegister_TSMSEProcedures(CL);
18575: SIRegister_TSMSEModule(CL);
18576: SIRegister_TSMSEModules(CL);
18577: CL.AddTypeS('TSMSScriptLanguage', '( slCustom, slVBScript, slJavaScript )');
18578: SIRegister_TSMSScriptExecutor(CL);
18579: //CL.AddDelphiFunction('Procedure Register');
18580: end;
18581:
18582: procedure SIRegister_geometry2(CL: TPSPPascalCompiler);
18583: begin
18584:   TPointF2', 'record X : Double; Y : Double; end');
18585:   TPoint3', 'record X : integer; Y : integer; Z : Integer; end');
18586: //CL.AddTypes('TPointF3', 'TPointF3 // will not work');
18587:   TPointF3', 'record X : double; Y : double; Z : Double; end');
18588:   Function AreLinesParallel( p1, p2, p3, p4 : TPoint ) : Boolean;');
18589:   Function AreLinesParallel1( p1, p2, p3, p4 : TPointF2; e : Double ) : Boolean;');
18590:   Function AreLinesParallel2( p1, p2, p3, p4, p5, p6 : TPoint3 ) : Boolean;');
18591:   Function AreLinesParallel3( p1, p2, p3, p4, p5, p6 : TPointF3; e : Double ) : Boolean;');
18592:   Function IntersectLines( p1, p2, p3, p4 : TPoint ) : TPoint;');
18593:   Function IntersectLines1( p1, p2, p3, p4 : TPointF2; e : Double ) : TPointF2;');
18594:   Function AngleDifference( alpha, beta : Double ) : Double');
18595: end;
18596:
18597: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
18598: New Functions /Classes mX4:
18599:
18600: function DownloadJPGToBitmap(const URL : string; ABitmap: TBitmap): Boolean;');
18601: procedure GetImageLinks(AURL: string; AList: TStrings);'
18602: procedure SaveByteCode;');
18603: procedure ResetKeyPressed;');
18604: procedure SetKeyPressed;');
18605:
18606: RIRegister_HTTPProd_Routines(Exec);
18607:   Function ContentFromScriptStream( AStream : TStream; AWebModuleContext : TWebModuleContext;
AStripParamQuotes : Boolean; AHandleTag : THHandleTagProc; AHandleScriptTag : THHandleScriptTag; const
AScriptEngine : string; '+
                           'ALocateFileService : ILocateFileService) : string');
18608:   Function ContentFromScriptFile( const AFileName : TFileName; AWebModuleContext : TWebModuleContext;
AStripParamQuotes : Boolean; AHandleTag : THHandleTagProc; AHandleScriptTag : THHandleScriptTag; '+
                           const AScriptEngine:string; ALocateFileService:ILocateFileService) : string');
18609:   Function FindComponentWebModuleContext( AComponent : TComponent ) : TWebModuleContext';
18610:   Function GetTagID( const TagString : string ) : TTag');
18611:   Function ContentFromStream( AStream : TStream; AStripParamQuotes : Boolean; AHandleTag : THHandleTagProc;
AHandledTag : THHandleTagProc ) : string');
18612:   Function ContentFromString( const AValue : string; AStripParamQuotes : Boolean; AHandleTag : THHandleTagProc;
AHandledTag : THHandleTagProc ) : string');
18613:   Function ContentFromAValue( const AValue : string; AStripParamQuotes : Boolean; AHandleTag : THHandleTagProc;
AHandledTag : THHandleTagProc ) : string');
18614:   Function ContentFromAValue( const AValue : string; AStripParamQuotes : Boolean; AHandleTag : THHandleTagProc;
AHandledTag : THHandleTagProc ) : string');
18615:
18616: RIRegister_synacrypt_Routines(Exec);
18617:   function TestDes: boolean;
18618:   (:Call internal test of all 3DES encryptions. Returns @true if all is OK.)
18619:   function Test3Des: boolean;
18620:   (:Call internal test of all AES encryptions. Returns @true if all is OK.)
18621:   function TestAes: boolean;
18622:
18623: Procedure LogMessage( const Fmt : string; const Params : array of const );
18624: Function UnixPathToDosPath2( const Path : string ) : string';
18625: Function DosPathToUnixPath2( const Path : string ) : string';
18626: Procedure InitISAPIApplicationList');
18627: Procedure DoneISAPIApplicationList');
18628:
18629: RIRegister_xmldom_Routines(Exec);
18630:   Function IsPrefixed( const AName : DOMString ) : Boolean';
18631:   Function IsPrefixedW( const AName : DOMStringW ) : Boolean');
18632:   Function ExtractLocalName( const AName : DOMString ) : DOMString');
18633:   Function ExtractLocalNameW( const AName : DOMStringW ) : DOMStringW');
18634:   Function ExtractPrefixW( const AName : DOMStringW ) : DOMStringW');
18635:   Function MakeNodeNameW( const Prefix, LocalName : DOMStringW ) : DOMStringW');
18636:   Function ExtractPrefix( const AName : DOMString ) : DOMString');
18637:   Function MakeNodeName( const Prefix, LocalName : DOMString ) : DOMString');
18638:   Function SameNamespace( const Node:IDOMNode,const namespaceURI:WideString):Boolean;');
18639:   Function SameNamespace2( const URI1, URI2 : WideString ) : Boolean');
18640:   Function NodeMatches( const Node:IDOMNode; const TagName, NamespaceURI:DOMString ) : Boolean');
18641:   Function GetDOMNodeEx( const Node : IDOMNode ) : IDOMNodeEx');
18642:   Procedure RegisterDOMVendor( const Vendor : TDOMVendor );
18643:   Procedure UnRegisterDOMVendor( const Vendor : TDOMVendor );
18644:   Function GetDOMVendor( VendorDesc : string ) : TDOMVendor');
18645:   Function GetDOM( const VendorDesc : string ) : IDOMImplementation');
18646:   Procedure DOMVendorNotSupported( const PropOrMethod, VendorName : string );
18647:   Function IsValidLocale( Locale : LCID; dwFlags : DWORD ) : BOOL');
18648:   Function ConvertDefaultLocale( Locale : LCID ) : LCID');
18649:   Function GetThreadLocale : LCID');
18650:   Function SetThreadLocale( Locale : LCID ) : BOOL');
18651:   Function GetSystemDefaultLangID : LANGID');
18652:   Function GetUserDefaultLangID : LANGID');
18653:   Function GetSystemDefaultLCID : LCID');
18654:   Function GetUserDefaultLCID : LCID');

```

```

18655:
18656: Function AbsInt( const B : integer ) : integer';
18657: Function AbsFloat( const B : double ) : extended';
18658:
18659: Procedure ReconcileDeltas( const cdsArray:array of TClientDataset; vDeltaArray: OleVariant );
18660: Procedure CDSApplyUpdates( ADatabase : TDatabase; var vDeltaArray : OleVariant; const vProviderArray : OleVariant; Local : Boolean );
18661: Function (@CheckMemory, 'procedure CheckMemory;');
18662: Function (@GetMemoryInfo, 'function getMemoryInfo;');
18663: Function (@GetMemoryInfo, 'function getMemInf;');
18664: Procedure RaiseLastWin32_2( const Text : string );
18665:
18666: RIRegister_Gameboard_Routines(Exec);
18667: Function Opponent( Player : TPlayer ) : TPlayer';
18668: Procedure InitZobritsNumbers( var ZobristNumbers, Count : Integer );
18669: Procedure SaveStringToStream( const Str : String; Stream : TStream );
18670: Function LoadStringFromStream( Stream : TStream ) : String';
18671: Function WaitForSyncObject( SyncObject:THandle;Timeout:Cardinal;BlockInput:Boolean ):Cardinal;
18672: Function ProcessMessage : Boolean';
18673: Procedure ProcessMessages( Timeout : DWORD ); //without application!
18674:
18675: RIRegister_PersistSettings_Routines(Exec);
18676: Procedure SetStorageHandler( AFunction : TStorageHandlerFunction );
18677: Procedure SetStorageHandler1( AMethod : TStorageHandlerMethod );
18678: Function GetStorage : TPersistStorage';
18679: Procedure SaveComponents( Root : TComponent; Storage : TPersistStorage );
18680: Procedure LoadComponents( Root : TComponent; Storage : TPersistStorage );
18681: Procedure AutoSave( Root : TComponent );
18682: Procedure AutoLoad( Root : TComponent );
18683:
18684: procedure FloatToDecimalE(var Result: TFLOATREC; const Value: extended; ValueType: TFLOATVALUE; Precision, Decimals: Integer );
18685: function FloatToTextE(BufferArg: PChar; const Value: extended; ValueType: TFLOATVALUE; Format: TFLOATFORMAT; Precision, Digits: Integer );
18686: procedure FloatToDecimalE(var Result: TFLOATREC; const Value: extended; ValueType: TFLOATVALUE; Precision, Decimals: Integer );
18687: function FloatToTextE(BufferArg: PChar; const Value: extended; ValueType: TFLOATVALUE; Format: TFLOATFORMAT; Precision, Digits: Integer );
18688: Function GetSystemDefaultLCID : LCID';
18689: Function GetUserDefaultLCID : LCID';
18690: Function CreateMutex2( lpMutexAttributes:TObject;bInitialOwner:BOOL;lpName:PChar ) : THandle';
18691: Function CreateSemaphore2( lpSemaphoreAttributes : TObject; lInitialCount, lMaximumCount : Longint; lpName : PChar ) : THandle';
18692:
18693: Function GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD ) : BOOL';
18694: Createmutex2 fhand:= OpenFileHandle(exepath+'maxbox4.exe')
18695:
18696: function getMatchString(arex, atext: string) : string;
18697: function getLastInput: DWord;
18698: procedure GetKLLIST(List: TStrings); //Keyboardlist2
18699: procedure EnableCTRLALTDEL(YesNo : boolean);
18700: function LocalIP: string;
18701: function IPAddrToName(IPAddr: string) : string;
18702: function GetIPFromHost(const HostName: string) : string;
18703: function FindComputers(Computers : TStringList) : DWORD;
18704: function GetWin32TypeLibList(var Lines: TStringList) : Boolean;
18705: function RecurseWin32(const R: TRegistry; const ThePath: string;
18706: const TheKey: string) : string;
18707: function RunAsAdmin(hWnd: HWND; filename: string; Parameters: string) : Boolean;
18708: procedure SaveGraphicToStream(Graphic: TGraphic; Stream: TStream);
18709: function LoadGraphicFromStream(Stream: TStream) : TGraphic;
18710: procedure CopyStreamToFile(S: TStream; F: THandle);
18711: function BigPow(aone, atwo: integer) : string;
18712: PROCEDURE GetPelsPerMeter(CONST Bitmap: TBitmap;
18713: VAR xPelsPerMeter, yPelsPerMeter: INTEGER);
18714: PROCEDURE RainbowColor(CONST fraction: Double; VAR R,G,B: BYTE);
18715:
18716: RIRegister_JclPCRE_Routines(S: TPSEexec);
18717: Procedure InitializeLocaleSupport';
18718: Procedure TerminateLocaleSupport';
18719: Function StrReplaceRegEx(const Subject, Pattern: AnsiString; Args: array of const):AnsiString;
18720:
18721: procedure SIRegister_VariantRtn(CL: TPSPascalCompiler);
18722: begin
18723: TProcReadElementValue', 'procedure(Value:Variant; IndexValue:integerarray; const HighBoundInd:integer; Var Continue:boolean)';
18724: TProcWriteElementValue', 'procedure(OldValue:Variant; IndexValue:integerarray; Var NewValue:Variant; Var Continue:boolean)';
18725:
18726: //TProcReadElementValue=procedure (Value:Variant; IndexValue:array of integer; const HighBoundInd:integer; Var Continue:boolean);
18727: //TProcWriteElementValue=procedure (OldValue:Variant; IndexValue:array of integer; Var NewValue:Variant; Var Continue:boolean);
18728: //
18729:
18730: Function SafeVarArrayCreate(const Bounds: array of Integer; VarType, DimCount: Integer) : Variant';
18731: Function VarArrayGet2(const A:Variant;const Indice:array of Integer;const HighBound:integer):Variant;
18732: Procedure VarArrayPut2( var A : Variant; const Value : Variant; const Indices : array of Integer; const HighBound : integer );

```

```

18733: Function CycleReadArray( vArray : Variant; CallBackProc : TProcReadElementValue) : boolean';
18734: Function CycleWriteArray(var vArray:Variant;CallBackProc: TProcWriteElementValue): boolean';
18735: Function CompareVarArray1( vArray1, vArray2 : Variant) : boolean';
18736: Function EasyCompareVarArray1(vArray1,vArray2:Variant;HighBound:integer): boolean';
18737: end;
18738:
18739: procedure SIRegister_StdFuncs(CL: TPSPascalCompiler);
18740: begin
18741:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EParserError');
18742:   //CL.AddTypeS('TCharSet', 'set of Char');
18743:   Function ConvertFromBase(sNum: String; iBase: Integer; cDigits: String): Integer';
18744:   Function ConvertToBase( iNum, iBase : Integer; cDigits : String) : String';
18745:   Function EnsureSentenceTerminates( Sentence : String; Terminator : Char) : String';
18746:   Function FindTokenStartingAt(st : String; var i: Integer; TokenChars: TCharSet; TokenCharsInToken : Boolean) : String';
18747:   Function GetDirectoryOfFile( FileName : String) : String';
18748:   Function GetDirOfFile( FileName : String) : String';
18749:   Function GetTempFile( FilePrefix : String) : String';
18750:   Function Icon2Bitmap( Icon : HICON) : HBITMAP';
18751:   Function Maxfib( n1, n2 : Integer) : Integer';
18752:   Function MaxD( n1, n2 : Double) : Double';
18753:   Function Minfib( n1, n2 : Integer) : Integer';
18754:   Function MinD( n1, n2 : Double) : Double';
18755:   Function RandomStringfib( iLength : Integer) : String';
18756:   Function RandomIntegerfib( iLow, iHigh : Integer) : Integer';
18757:   Function Soundexfib( st : String) : String';
18758:   Function StripStringfib( st : String; CharsToStrip : String) : String';
18759:   Function ClosestWeekday( const d : TDateTime) : TDateTime';
18760:   Function Yearfib( d : TDateTime) : Integer';
18761:   Function Monthfib( d : TDateTime) : Integer';
18762:   Function DayOffYearfib( d : TDateTime) : Integer';
18763:   Function DayOfMonth( d : TDateTime) : Integer';
18764:   Function VarCoalesce( V1, V2 : Variant) : Variant';
18765:   Function VarEqual( V1, V2 : Variant) : Boolean';
18766:   Procedure WeekOfYearfib( d : TDateTime; var Year, Week : Integer)';
18767:   Function Degree10( Degree : integer) : double';
18768:   Function CompToStr( Value : comp) : string';
18769:   Function StrToComp( const Value : string) : comp';
18770:   Function CompDiv( A, B : comp) : comp';
18771:   Function CompMod( A, B : comp) : comp';
18772:   // CL.AddTypeS('PComp', '^Comp // will not work');
18773: end;
18774:
18775: procedure SIRegister_RegUtils(CL: TPSPascalCompiler);
18776: begin
18777:   Procedure DefWriteToRegistry(const OtherKeys,ParamNams:array of string;const Values:array of Variant);
18778:   Procedure WriteToRegistry( aRootKey : HKEY; const OtherKeys, ParamNames : array of string; const Values : array of Variant)';
18779:   Function ReadFromRegistry(aRootKey : HKEY; const OtherKeys, ParamNames: array of string): Variant';
18780:   Function DefReadFromRegistry( const OtherKeys, ParamNames : array of string) : Variant';
18781:   Function AllSubKey( aRootKey : HKEY; const ForPath : array of string) : Variant';
18782:   Function DefAllSubKey( const ForPath : array of string) : Variant';
18783:   Function SaveRegKey( const FileName : String; const ForKey : array of string) : Boolean';
18784:   Function LoadRegKey( const FileName : String; const ForKey : array of string) : Boolean';
18785:   Function AltSaveRegKey( const FileName : String; const ForKey : array of string) : Boolean';
18786:   Function AltLoadRegKey( const FileName : String; const ForKey : array of string) : Boolean';
18787:   Function GetKeyForParValue( const aRootKey, ParName, ParValue : string) : string';
18788: end;
18789:
18790:   Function GetEulerPhi( n : int64) : int64';
18791:   function isprime(f: int64): boolean;';
18792:   function DispositionFrom(const SQLText:string):TPoint;
18793:   procedure AllTables(const SQLText:string;FTables:Tstrings);
18794:   function TableByAlias(const SQLText,Alias:string):string;
18795:   function FullFieldName(const SQLText,FieldName:string):string;
18796:   function AddToWhereClause(const SQLText,NewClause:string):string;
18797:   function GetWhereClause(SQLText:string;N:integer;var StartPos,EndPos:integer ):string;
18798:
18799:   function WhereCount (SQLText:string):integer;
18800:   function GetOrderInfo (SQLText:string):variant;
18801:   function OrderStringTxt (SQLText:string; var StartPos,EndPos:integer ):String;
18802:   function PrepareConstraint (Src:Tstrings):string;
18803:   procedure DeleteEmptyStr (Src:Tstrings);
18804:   function NormalizeSQLText (const SQL: string;MacroChar:Char): string;
18805:   function CountSelect (const SrcSQL:string):string;
18806:   function GetModifyTable (const SQLText:string;AlreadyNormal:boolean):string;
18807:   function GetCharFromVKey(vkey: Word): string;
18808:   function Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile: string): Boolean;
18809:   function IsObjectActive (ClassName: string): Boolean;
18810:   function GetActiveObject (ClassID:TGUID; anil:TObject; aUnknown:IUnknown):HRESULT;';
18811:   function RegisterOCX (FileName: string): Boolean;
18812:   function UnRegisterOCX (FileName: string): Boolean;
18813:   function RegisterServer2 (const aDllFileName: string; aRegister: Boolean): Boolean;
18814:   procedure mIRCDE (Service, Topic, Cmd: string);
18815:   //mIRCDE('mIRC', 'COMMAND', '/say Hallo von SwissDelphiCenter.ch');
18816:   function OpenIE (aURL: string): boolean;
18817:   function XRTLIsInMainThread: Boolean;
18818:   function IsInMainThread: Boolean;
18819:

```

```

18820: procedure SIRегистер_SpectraLibrary(CL: TPSPPascalCompiler);
18821: begin
18822:   CL.AddTypeS('Nanometers', 'Double');
18823:   CL.AddConstantN('WavelengthMinimum','LongInt').SetInt( 380);
18824:   CL.AddConstantN('WavelengthMaximum','LongInt').SetInt( 780);
18825:   Procedure WavelengthToRGB( const Wavelength : Nanometers; var R, G, B : BYTE );
18826: end;
18827: procedure SIRегистер_DrawFigures(CL: TPSPPascalCompiler);
18828: begin
18829:   Procedure DrawCube( const PantoGraph : TPantoGraph; const color : TColor );
18830:   Procedure DrawSphere( const PantoGraph : TPantoGraph; const LatitudeColor, LongitudeColor : TColor; const
LatitudeCircles, LongitudeSemicircles, PointsInCircle : WORD );
18831:   Procedure DrawSurface( const PantoGraph : TPantoGraph );
18832:   Procedure DrawFootballField(const PantoGraph:TPantoGraph;const ColorField,ColorLetters,ColorGoals:
TColor );
18833: end;
18834: procedure SIRегистер_synadbg(CL: TPSPPascalCompiler);
18835: begin
18836:   SIRегистер_TSynaDebug(CL);
18837:   Procedure AppendToLog( const value : Ansistring );
18838: end;
18839:
18840: procedure SIRегистер_XmlRpcCommon(CL: TPSPPascalCompiler);
18841: begin
18842:   SIRегистер_TRC4(CL);
18843:   CL.AddTypeS('TRPCDataType', '( rpNone, rpString, rpInteger, rpBoolean, rpDoub'
18844:     +'le, rpDate, rpBase64, rpStruct, rpArray, rpName, rpError )');
18845:   Function GetTempDirRPC : string';
18846:   Function FileIsExpired( const FileName : string; Elapsed : Integer ) : Boolean';
18847:   Function EncodeEntities( const Data : string ) : string';
18848:   Function DecodeEntities( const Data : string ) : string';
18849:   Function ReplaceRPC(const Data: string; const Find: string; const Replace: string): string';
18850:   Function InStr( Start : Integer; const Data : string; const Find : string ) : Integer';
18851:   Function Mid( const Data : string; Start : Integer ) : string';
18852:   Function DateTimeToISO( ConvertDate : TDateTime ) : string';
18853:   Function IsoDateTime( const ISOStringDate : string ) : TDateTime';
18854:   Function ParseStringRPC( const SearchString : string; Delimiter : Char; Substrings : TStrings; const
AllowEmptyStrings : Boolean; ClearBeforeParse : Boolean ) : Integer';
18855:   Function ParseStream( SearchStream : TStream; Delimiter : Char; Substrings : TStrings; AllowEmptyStrings
: Boolean; ClearBeforeParse : Boolean ) : Integer';
18856:   Function FixEmptyString( const Value : string ) : string';
18857:   Function URLEncodeRPC( const Value : string ) : string';
18858:   Function StreamToStringRPC( Stream : TStream ) : string';
18859:   Procedure StringToStream( const Text : string; Stream : TStream );
18860:   Function StreamToVariant( Stream : TStream ) : OleVariant';
18861:   Procedure VariantToStream( V : OleVariant; Stream : TStream );
18862:   Function Hash128AsHex( const Hash128Value : T4x4longWordRecord ) : string';
18863:   CL.AddConstantN('ValidURLChars','String').SetString( 'ABCDEFIGHJKLMNOPQRSTUVWXYZ0123456789$-_@.&+-!''*'(),'
;#/?:' );
18864: end;
18865:
18866: procedure SIRегистер_synafpc(CL: TPSPPascalCompiler);
18867: begin
18868:   Function LoadLibraryfpc( ModuleName : PChar ) : TLibHandle';
18869:   Function FreeLibraryfpc( Module : TLibHandle ) : LongBool';
18870:   Function GetProcAddressfpc( Module : TLibHandle; Proc : PChar ) : Pointer';
18871:   Function GetModuleFileNamefpc(Module:TLibHandle;Buffer:PChar; BufLen:Integer):Integer;
18872:   CL.AddTypeS('TLibHandle', 'Integer');
18873:   CL.AddTypeS('TLibHandle2', 'HModule');
18874:   CL.AddTypeS('LongWordfpc', 'DWord');
18875:   Procedure Sleepfpc( milliseconds : Cardinal );
18876: end;
18877:
18878: procedure SIRегистер_Spring_Utilsmx(CL: TPSPPascalCompiler);
18879: begin
18880:   CL.AddTypeS('TOSPlatformType', '( ptUnknown, ptWin3x, ptWin9x, ptWinNT )');
18881:   SIRегистер_TOperatingSystem(CL);
18882:   SIRегистер_TEnvironmentClass(CL);
18883:   CL.AddTypeS('Environment', 'TEnvironment');
18884:   Function ApplicationPath : string';
18885:   Function ApplicationVersion : TVersion';
18886:   Function ApplicationVersionString : string';
18887:   Function GetLastErrorMessage : string';
18888:   Function CreateCallback( obj : TObject; methodAddress : Pointer ) : TCallbackFunc';
18889:   Function ConvertFileTimeToDate( const fileTime:TFfileTime;useLocalTimeZone:Bool ):TDateTime';
18890:   Function ConvertDateTimeToFileTime( const datetime:TDateTime; useLocalTimeZone:Bool ):TFfileTime;
18891:   {Procedure Synchronize( threadProc : TThreadProcedure );
18892:   Procedure Queue( threadProc : TThreadProcedure ); }
18893:   Function TryGetPropInfo(instance:TObject;const propertyName:string;out propInfo:PPropInfo):Bool;
18894:   Function IsCtrlPressed : Boolean';
18895:   Function IsShiftPressed : Boolean';
18896:   Function IsAltPressed : Boolean';
18897:   Procedure CheckFileExists( const fileName : string );
18898:   Procedure CheckDirectoryExists( const directory : string );
18899:   CL.AddConstantN('COneKB','Int64').SetInt64( 1024 );
18900:   CL.AddConstantN('COneMB','Int64').SetInt64( 1048576 );
18901:   CL.AddConstantN('COneGB','Int64').SetInt64( 1073741824 );
18902:   CL.AddConstantN('COneTB','Int64').SetInt64( 1099511627776 );
18903:   function TryConvertStrToDate( const s,format:string;out value:TDateTime ):Boolean;

```

```

18904: function ConvertStrToDateTIme(const s, format: string): TDateTime;');
18905: end;
18906:
18907: procedure SIRegister_rxOle2Auto(CL: TPSPPascalCompiler);
18908: begin
18909:   CL.AddConstantN('MaxDispArgs','LongInt').SetInt( 64);
18910:   CL.AddConstantN('MaxDispArgs','LongInt').SetInt( 32);
18911:   CL.AddClassN(CL.FindClass('TOBJECT'),'EPropReadOnly');
18912:   CL.AddClassN(CL.FindClass('TOBJECT'),'EPropWriteOnly');
18913:   SIRegister_ToleController(CL);
18914:   Procedure InitOLE2';
18915:   Procedure DoneOLE1';
18916:   Function OleInitialized : Boolean)';
18917:   Function MakeLangID( PrimaryLangID, SubLangID : Word) : Word';
18918:   Function MakeLCID( LangID : Word) : TLCID';
18919:   Function CreateLCID( PrimaryLangID, SubLangID : Word) : TLCID';
18920:   Function ExtractLangID( LCID : TLCID) : Word';
18921:   Function ExtractSubLangID( LCID : TLCID) : Word';
18922: end;
18923:
18924: procedure SIRegister_ologifit(CL: TPSPPascalCompiler);
18925: begin
18926:   Procedure LogiFit( X, Y : TVector; Lb, Ub : Integer; ConsTerm : Boolean; General : Boolean; MaxIter : Integer; Tol : Float; B : TVector; V : TMatrix)';
18927:   Procedure WLogiFit( X, Y, S : TVector; Lb, Ub : Integer; ConsTerm : Boolean; General : Boolean; MaxIter : Integer; Tol : Float; B : TVector; V : TMatrix)';
18928:   Function LogiFit_Func( X : Float; B : TVector) : Float';
18929: end;
18930:
18931: Procedure FormattedTextOut(TargetCanvas: TCanvas; const Rect : TRect; const Text : string; Selected : Boolean; Columns : TProposalColumns; Images : TImageList)';
18932: Function FormattedTextWidth( TargetCanvas : TCanvas; const Text : string; Columns : TProposalColumns; Images : TImageList) : Integer';
18933: Function PrettyTextToFormattedString(const APrettyText: string; AlternateBoldStyle:Boolean):string;
18934:
18935: procedure SIRegister_uLinfit(CL: TPSPPascalCompiler);
18936: begin
18937:   Procedure LinFit( X, Y : TVector; Lb, Ub : Integer; B : TVector; V : TMatrix)';
18938:   Procedure WLInFit( X, Y, S : TVector; Lb, Ub : Integer; B : TVector; V : TMatrix)';
18939:   Procedure SVLDLinFit(X,Y:TVector; Lb,Ub:Integer; SVDTol : Float; B: TVector; V : TMatrix)';
18940:   Procedure WSVDLinFit(X,Y,S:TVector; Lb,Ub:Integer; SVDTol : Float; B: TVector; V : TMatrix)';
18941:   Procedure SVDFit( X : TMatrix; Y : TVector; Lb, Ub, Nvar : Integer; ConsTerm : Boolean; SVDTol : Float; B : TVector; V : TMatrix)';
18942:   Procedure WSVDFit( X : TMatrix; Y, S : TVector; Lb, Ub, Nvar : Integer; ConsTerm : Boolean; SVDTol : Float; B : TVector; V : TMatrix)';
18943: end;
18944:
18945: procedure SIRegister_MaxUtils(CL: TPSPPascalCompiler);
18946: begin
18947:   CL.AddTypeS('MaxCharSet', 'set of Char');
18948:   Function GetMachineNameMax : String';
18949:   Function GetModuleNameMax( HModule : THandle) : String';
18950:   Function TrimChars( const S : string; Chars : MaxCharSet) : string';
18951:   Function TickCountToDateTIme( Ticks : Cardinal) : TDateTime';
18952:   Procedure OutputDebugStringMax( const S : String)';
18953:   Procedure OutputDebugFormat( const FmtStr : String; Args : array of const)';
18954:   Function IsAppRunningInDelphi : Boolean';
18955:   Procedure ParseFields(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strings:TStrings;Decode: Bool;
18956:   Function HTTPDecodemax( const AStr : String) : string';
18957:   Function HTTPEncodemax( const AStr : String) : string';
18958:   Function FormatDate( const DateString : string) : string';
18959:   Function FormatListMasterDate(const DateStr,FormatDefStr:String; Len:Integer):String';
18960:   Function InvertCase( const S : String) : String';
18961:   Function CommentLinesWithSlashes( const S : String) : String';
18962:   Function UncommentLinesWithSlashes( const S : String) : String';
18963:   Function StripChars( const S : String; Strip : CharSet) : String';
18964:   Function TrimChars( const S : string; Chars : CharSet) : string';
18965:   Function TrimLeftChars( const S : string; Chars : CharSet) : string';
18966:   Function TrimRightChars( const S : string; Chars : CharSet) : string';
18967:   Function ContainsChars( const S : String; Strip : CharSet) : Boolean';
18968:   Function DequotedStrmax( const S : String; AQuoteChar : Char) : String';
18969:   Procedure LeftPadStr( var S : String; toLength : Integer; withChar : Char)';
18970:   Procedure RightPadStr( var S : String; toLength : Integer; withChar : Char)';
18971:   Function RemoveChars( S : string; Chars : CharSet) : string';
18972:   Function FilterChars( S : string; Chars : CharSet) : string';
18973:   Function RemoveNonNumericChars( S : string) : string';
18974:   Function RemoveNonAlphanumChars( S : string) : string';
18975:   Function RemoveNonAlphaChars( S : string) : string';
18976:   Function HasAlphaChars( S : string) : boolean';
18977:   Function ReplaceChars( S : string; Chars : CharSet; ReplaceWith : Char) : string';
18978:   Function DomainOfEMail( const EMailAddress : String) : String';
18979:   Function IPToHexIP( const IP : String) : String';
18980:   Procedure CmdLineToStrings( S : AnsiString; const List : TStrings)';
18981:   BASE2','String').SetString( '01';
18982:   CL.AddConstantN('BASE10','String').SetString( '0123456789');
18983:   CL.AddConstantN('BASE16','String').SetString( '0123456789ABCDEF');
18984:   CL.AddConstantN('BASE36','String').SetString( '0123456789ABCDEFGHijklmnopqrstuvwxyz');
18985:   CL.AddConstantN('BASE62','String').SetString(
  '0123456789ABCDEFGHijklmnopqrstuvwxyz');

```

```

18986: Function BaseConvert( Number, FromDigits, ToDigits : String ) : String';
18987: Function ValidXmlName( AName : PChar; ASize : Integer ) : Boolean;';
18988: Function ValidXmlName1( const AName : String ) : Boolean;');
18989: Function EncodeXmlAttributeValue( const AStr : AnsiString ) : AnsiString;';
18990: Procedure EncodeXmlAttributeValue3(ABuff : PChar; ABuffSize : Integer; var AStr : AnsiString; var ALen,
ALenOffset : Integer);');
18991: Procedure EncodeXmlAttributeValue4( const ASource: String; var ADest: String; var ALen, AOffset: Integer );
18992: Function EncodeXmlAttribute5( const AStr : String ) : String;';
18993: Procedure EncodeXmlAttribute6(ABuff:PChar;ABuffSize:Int;var AStr:AnsiString;var ALen,AOffset:Int;
18994: Procedure EncodeXmlComment( const ASource:AnsiString;var ADest:AnsiString;var ALen,AOffset:Integer );
18995: Function HasEncoding( const AStr : AnsiString ) : Boolean;');
18996: Function DecodeXmlAttributeValue( const AStr : String ) : String');
18997: Procedure ReallocateString(var AString:AnsiString; var ALen : Integer; AReqLen : Integer)');
18998: //Procedure AttrFillXMLString( AnAttr : IAttribute; var aString : AnsiString; var aOffset, aLen :
ALen );
18999: //Procedure FillXMLString( ANode : INode; var AString : String; var AOffset, ALen : Integer; ASibling :
INode; ALevel : Integer );
19000: //Function NodeToXML( ANode : INode ) : String';
19001: //Procedure XMLSaveToFile( ANode : INode; const AFileName : String );
19002: //Function XMLLoadFromFile( const AFileName : String ) : INode';
19003: Function Hashmax( const ASource : AnsiString ) : Cardinal';
19004: CL.AddConstantN('GXMLIndentSpaces','Integer').SetInt( 2 );
19005: //CL.AddConstantN('GXMLMultiLineAttributes','Boolean')BoolToStr( True );
19006: GXMLMultiLineAttributeThreshold','Integer').SetInt( 7 );
19007: end;
19008:
19009: procedure SIRегистre_MaxDOMDictionary(CL: TPSPascalCompiler);
19010: begin
19011:   SIRегистre_IDictionary(CL);
19012:   SIRегистre_TDictionary(CL);
19013:   Function HashFast( const AKey : String ) : Cardinal';
19014:   Function HashCarlos( const AKey : String ) : Cardinal';
19015:   Function BorlandHashOf( const AKey : String ) : Cardinal';
19016:   Function HashSumOfChars( const AKey : String ) : Cardinal';
19017: end;
19018:
19019: procedure SIRегистre_MaxDOM(CL: TPSPascalCompiler);
19020: begin
19021:   CL.AddTypeS('TNodeType', '( ntElement, ntText, ntCDATA, ntComment )');
19022:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),INode,'INode');
19023:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IAttribute,'IAttribute');
19024:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IAttributeCollection,'IAttributeCollection');
19025:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),INodeCollection,'INodeCollection');
19026:   SIRегистre_INode(CL);
19027:   SIRегистre_IAttribute(CL);
19028:   SIRегистre_IAttributeCollection(CL);
19029:   SIRегистre_INodeCollection(CL);
19030:   Function NodeCreate( const ANodeName : String; ANodeType : TNodeType ) : INode';
19031:   SIRегистre_TAttribute(CL);
19032:   CL.AddTypeS('TNodes', 'array of INode');
19033:   CL.AddTypeS('TAttributes2', 'array of IAttribute');
19034:   CL.AddTypeS('THashedAttributes', 'record Attributes : TAttributes2; AttrCount :
19035:     +: Integer; AttrCapacity : Integer; end');
19036:   CL.AddTypeS('TAttrHashTable', 'array of THashedAttributes');
19037:   SIRегистre_TNode(CL);
19038: //CL.AddTypeS('TNodeClass', 'class of TNode');
19039: //CL.AddTypeS('TAttributeClass', 'class of TAttribute');
19040:   Function PointerToStr( P : __Pointer ) : String';
19041:   Function StrToPointer( const S : String ) : __Pointer';
19042:   Function INodeToStr( ANode : INode ) : String';
19043:   Function StrToINode( const S : String ) : INode';
19044:   Function CompareByNameNode( N1, N2 : INode ) : Integer';
19045:   Function CompareByNameAttr( N1, N2 : INode ) : Integer';
19046: end;
19047:
19048: procedure SIRегистre_cASN1(CL: TPSPascalCompiler);
19049: begin
19050:   CL.AddClassN(CL.FindClass('TOBJECT'),'EASN1');
19051:   'ASN1_ID_END_OF_CONTENT','LongWord').SetUInt( $00 );
19052:   CL.AddConstantN('ASN1_ID_BOOLEAN','LongWord').SetUInt( $01 );
19053:   CL.AddConstantN('ASN1_ID_INTEGER','LongWord').SetUInt( $02 );
19054:   CL.AddConstantN('ASN1_ID_BIT_STRING','LongWord').SetUInt( $03 );
19055:   'ASN1_ID_OCTET_STRING','LongWord').SetUInt( $04 );
19056:   'ASN1_ID_NULL','LongWord').SetUInt( $05 );
19057:   'ASN1_ID_OBJECT_IDENTIFIER','LongWord').SetUInt( $06 );
19058:   'ASN1_ID_OBJECT_DESCRIPTOR','LongWord').SetUInt( $07 );
19059:   'ASN1_ID_EXTERNAL','LongWord').SetUInt( $08 );
19060:   'ASN1_ID_REAL','LongWord').SetUInt( $09 );
19061:   'ASN1_ID_ENUMERATED','LongWord').SetUInt( $0A );
19062:   'ASN1_ID_EMBEDDED_PDV','LongWord').SetUInt( $0B );
19063:   'ASN1_ID_UTF8STRING','LongWord').SetUInt( $0C );
19064:   'ASN1_ID_RELATIVE_OID','LongWord').SetUInt( $0D );
19065:   'ASN1_ID_NUMERICSTRING','LongWord').SetUInt( $12 );
19066:   'ASN1_ID_PRINTABLESTRING','LongWord').SetUInt( $13 );
19067:   'ASN1_ID_T61STRING','LongWord').SetUInt( $14 );
19068:   'ASN1_ID_VIDEOTEXSTRING','LongWord').SetUInt( $15 );
19069:   'ASN1_ID_IA5STRING','LongWord').SetUInt( $16 );
19070:   'ASN1_ID_UTCTIME','LongWord').SetUInt( $17 );
19071:   'ASN1_ID_GENERALIZEDTIME','LongWord').SetUInt( $18 );

```

```

19072:   'ASN1_ID_GRAPHICSTRING', 'LongWord').SetUInt( $19);
19073:   'ASN1_ID_VISIBLESTRING', 'LongWord').SetUInt( $1A);
19074:   'ASN1_ID_GENERALSTRING', 'LongWord').SetUInt( $1B);
19075:   'ASN1_ID_UNIVERSALSTRING', 'LongWord').SetUInt( $1C);
19076:   'ASN1_ID_CHARACTERSTRING', 'LongWord').SetUInt( $1D);
19077:   'ASN1_ID_BMPSTRING', 'LongWord').SetUInt( $1E);
19078:   'ASN1_ID_SEQUENCE', 'LongWord').SetUInt( $30);
19079:   'ASN1_ID_SET', 'LongWord').SetUInt( $31);
19080:   'ASN1_ID_CONSTRUCTED', 'LongWord').SetUInt( $20);
19081:   'ASN1_ID_APPLICATION', 'LongWord').SetUInt( $40);
19082:   'ASN1_ID_CONTEXT_SPECIFIC', 'LongWord').SetUInt( $80);
19083:   'ASN1_ID_PRIVATESET', 'LongWord').SetUInt( $C0);
19084:   TASN1ObjectIdentifier', 'array of Integer');
19085:   TASN1ParseProc', 'procedure (const TypeID: Byte; const DataBuf: string; const DataSize: Integer; const ObjectIdx: Integer; const CallerData: Integer);');

19086:
19087: {TASN1ParseProc =
19088:   procedure (const TypeID: Byte; const DataBuf: string; const DataSize: Integer;
19089:   const ObjectIdx: Integer; const CallerData: Integer); }

19090:
19091: Procedure ASN1OIDInit( var A : TASN1ObjectIdentifier; const B : array of Integer)');
19092: Function ASN1OIDToStr( const A : TASN1ObjectIdentifier) : AnsiString');
19093: Function ASN1OIDEqual( const A:TASN1ObjectIdentifier; const B : array of Integer) : Boolean';
19094: Function ASN1EncodeLength( const Len : Integer) : AnsiString');
19095: Function ASN1EncodeObj( const TypeID : Byte; const Data : AnsiString) : AnsiString');
19096: Function ASN1EncodeEndOfContent : AnsiString');
19097: Function ASN1EncodeNull : AnsiString');
19098: Function ASN1EncodeBoolean( const A : Boolean) : AnsiString');
19099: Function ASN1EncodeDataInteger8( const A : ShortInt) : AnsiString');
19100: Function ASN1EncodeDataInteger16( const A : SmallInt) : AnsiString');
19101: Function ASN1EncodeDataInteger24( const A : LongInt) : AnsiString');
19102: Function ASN1EncodeDataInteger32( const A : LongInt) : AnsiString');
19103: Function ASN1EncodeDataInteger64( const A : Int64) : AnsiString');
19104: Function ASN1EncodeInteger8( const A : ShortInt) : AnsiString');
19105: Function ASN1EncodeInteger16( const A : SmallInt) : AnsiString');
19106: Function ASN1EncodeInteger24( const A : LongInt) : AnsiString');
19107: Function ASN1EncodeInteger32( const A : LongInt) : AnsiString');
19108: Function ASN1EncodeInteger64( const A : Int64) : AnsiString');
19109: Function ASN1EncodeIntegerBuf( const A : string; const Size : Integer) : AnsiString');
19110: Function ASN1EncodeIntegerBufStr( const A : AnsiString) : AnsiString');
19111: Function ASN1EncodeEnumerated( const A : Int64) : AnsiString');
19112: Function ASN1EncodeBitString( const A : AnsiString; const UnusedBits : Byte) : AnsiString');
19113: Function ASN1EncodeOctetString( const A : AnsiString) : AnsiString');
19114: Function ASN1EncodeInt32AsOctetString( const A : LongInt) : AnsiString');
19115: Function ASN1EncodeUTF8String( const A : AnsiString) : AnsiString');
19116: Function ASN1EncodeIA5String( const A : AnsiString) : AnsiString');
19117: Function ASN1EncodeVisibleString( const A : AnsiString) : AnsiString');
19118: Function ASN1EncodeNumericString( const A : AnsiString) : AnsiString');
19119: Function ASN1EncodePrintableString( const A : AnsiString) : AnsiString');
19120: Function ASN1EncodeTeletexString( const A : AnsiString) : AnsiString');
19121: Function ASN1EncodeUniversalString( const A : WideString) : AnsiString');
19122: Function ASN1EncodeBMPString( const A : WideString) : AnsiString');
19123: Function ASN1EncodeUTCTime( const A : TDateTime) : AnsiString');
19124: Function ASN1EncodeGeneralizedTime( const A : TDateTime) : AnsiString');
19125: Function ASN1EncodeOID( const OID : array of Integer) : AnsiString');
19126: Function ASN1EncodeSequence( const A : AnsiString) : AnsiString');
19127: Function ASN1EncodeSet( const A : AnsiString) : AnsiString');
19128: Function ASN1EncodeContextSpecific( const I : Integer; const A : AnsiString) : AnsiString');
19129: Function ASN1DecodeLength(const Buf: string;const Size:Integer;var Len:Integer) : Integer');
19130: Function ASN1DecodeObjHeader( const Buf : string; const Size : Integer; var TypeID : Byte; var Len : Integer; var Data : string) : Integer');
19131: Function ASN1TypeIsConstructedType( const TypeID : Byte) : Boolean');
19132: Function ASN1TypeIsContextSpecific( const TypeID : Byte; var Idx : Integer) : Boolean');
19133: Function ASN1DecodeDataBoolean(const Buf:string;const Size:Integer; var A:Boolean) : Integer');
19134: Function ASN1DecodeDataInteger32(const Buf:string;const Size:Integer;var A:LongInt) : Integer');
19135: Function ASN1DecodeDataInteger64(const Buf:string; const Size:Integer; var A:Int64) : Integer');
19136: Function ASN1DecodeDataIntegerBuf(const Buf:string; const Size:Integer;var A : AnsiString) : Integer');
19137: Function ASN1DecodeDataBitString( const Buf : string; const Size : Integer; var A : AnsiString; var UnusedBits : Byte) : Integer');
19138: Function ASN1DecodeDataRawAnsiString(const Buf:string; const Size:Integer;var A:AnsiString) : Integer;
19139: Function ASN1DecodeDataOctetString(const Buf : string; const Size:Integer;var A:AnsiString) : Integer');
19140: Function ASN1DecodeDataIA5String(const Buf:string; const Size:Integer; var A:AnsiString) : Integer;
19141: Function ASN1DecodeDataVisibleString(const Buf:string;const Size:Integer;var A:AnsiString) : Integer;
19142: Function ASN1DecodeDataNumericString(const Buf:string;const Size:Integer;var A:AnsiString) : Integer;
19143: Function ASN1DecodeDataPrintableString(const Buf:string;const Size:Integer;var A:AnsiString) : Integer;
19144: Function ASN1DecodeDataTeletexString(const Buf:string;const Size:Integer; var A:AnsiString) : Integer;
19145: Function ASN1DecodeDataUTF8String(const Buf:string; const Size:Integer;var A:AnsiString) : Integer;
19146: Function ASN1DecodeDataUniversalString(const Buf:string;const Size:Integer;var A:AnsiString) : Integer;
19147: Function ASN1DecodeDataBMPString(const Buf : string; const Size:Integer; var A : AnsiString) : Integer;
19148: Function ASN1DecodeDataOID(const Buf:string;const Size:Integer;var A:TASN1ObjectIdentifier) : Integer;
19149: Function ASN1DecodeDataUTCTime(const Buf : string; const Size : Integer;var A : TDateTime) : Integer;
19150: Function ASN1DecodeDataGeneralizedTime(const Buf:string;const Size:Integer;var A:TDateTime) : Integer;
19151: Function ASN1Parse( const Buf : string; const Size : Integer; const ParseProc : TASN1ParseProc; const CallerData : Integer) : Integer';
19152: Function ASN1DecodeBoolean(const TypeID:Byte;const DataBuf:string;const DataSize:Int;var A:Bool) : Int;
19153: Function ASN1DecodeInteger32( const TypeID : Byte; const DataBuf : string; const DataSize : Integer; var A : LongInt) : Integer');
19154: Function ASN1DecodeInteger64( const TypeID : Byte; const DataBuf : string; const DataSize : Integer; var A : Int64) : Integer');

```

```

19155: Function ASN1DecodeIntegerBuf( const TypeID : Byte; const DataBuf : string; const DataSize : Integer; var
19156:   A : AnsiString) : Integer');
19157: Function ASN1DecodeBitString( const TypeID : Byte; const DataBuf : string; const DataSize : Integer; var
19158:   A : AnsiString; var UnusedBits : Byte) : Integer');
19159: Function ASN1DecodeString( const TypeID : Byte; const DataBuf : string; const DataSize : Integer; var A : 
19160:   AnsiString) : Integer');
19161: Function ASN1DecodeOID( const TypeID : Byte; const DataBuf : string; const DataSize : Integer; var A : 
19162:   TASN1ObjectIdentifier) : Integer');
19163: Function ASN1DecodeTime( const TypeID : Byte; const DataBuf : string; const DataSize : Integer; var A : 
19164:   TDateTime) : Integer');
19165: Procedure SelfTestASN1');
19166: end;
19167:
19168:
19169:
19170:   + ' Value : TX509AttributeValue; _Decoded : Boolean; end');
19171: //CL.AddTypeS('PX509AttributeTypeAndValue', '^TX509AttributeTypeAndValue // wil not work');
19172: Procedure InitX509AttributeTypeAndValue( var A : TX509AttributeTypeAndValue; const ATType : 
19173: TX509AttributeType; const Value : TX509AttributeValue)');
19174: Procedure InitX509AttributeName( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19175: Procedure InitX509AtSurname( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19176: Procedure InitX509AtGivenName( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19177: Procedure InitX509AtInitials( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19178: Procedure InitX509AtGenerationQualifier( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19179: Procedure InitX509AtCommonName( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19180: Procedure InitX509AtLocalityName(var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19181: Procedure InitX509AtStateOrProvince(var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19182: Procedure InitX509AtOrganizationName(var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19183: Procedure InitX509AtOrganizationUnitName(var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19184: Procedure InitX509AtTitle( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19185: Procedure InitX509AtDnQualifier( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19186: Procedure InitX509AtCountryName( var A : TX509AttributeTypeAndValue; const B : AnsiString)');
19187: Function EncodeX509AttributeTypeAndValue(const A : TX509AttributeTypeAndValue) : AnsiString');
19188: Function DecodeX509AttributeTypeAndValue( const TypeID : Byte; const Buf : string; const Size : Integer;
19189:   var A : TX509AttributeTypeAndValue) : Integer');
19190: // CL.AddTypeS('PX509RelativeDistinguishedName', '^TX509RelativeDistinguishedName // will not work');
19191: Procedure AppendX509RelativeDistinguishedName( var A : TX509RelativeDistinguishedName; const V : 
19192: TX509AttributeTypeAndValue)');
19193: Function EncodeX509RelativeDistinguishedName(const A : TX509RelativeDistinguishedName) : AnsiString');
19194: Function DecodeX509RelativeDistinguishedName( const TypeID : Byte; const Buf : string; const Size : 
19195: Integer; var A : TX509RelativeDistinguishedName) : Integer');
19196: CL.AddTypeS('TX509RDNSequence', 'array of TX509RelativeDistinguishedName');
19197: //CL.AddTypeS('PX509RDNSequence', '^TX509RDNSequence // will not work');
19198: Procedure AppendX509RDNSequence(var A : TX509RDNSequence; const B : TX509RelativeDistinguishedName);
19199: Function EncodeX509RDNSequence( const A : TX509RDNSequence) : AnsiString');
19200: Function DecodeX509RDNSequence( const TypeID : Byte; const Buf : string; const Size : Integer; var A : 
19201: TX509RDNSequence) : Integer';
19202: Function EncodeX509Name( const A : TX509Name) : AnsiString');
19203: Function DecodeX509Name(const TypeID : Byte; const Buf : string; const Size : Int; var A : TX509Name) : Integer;
19204:   'CurrentX509Version', 'string').SetString(' X509v3');
19205: type
19206:   TX509Version = (
19207:     X509v1 = 0,
19208:     X509v2 = 1,
19209:     X509v3 = 2,
19210:     X509vUndefined = $FF // implementation defined value );
19211:   TX509Version', '(X509v1, X509v2, X509v3, X509vUndefined)');
19212: type
19213:   TX509GeneralNameType = (
19214:     gnOtherName          = 0,
19215:     gnRFC822Name         = 1,
19216:     gnDNSName            = 2,
19217:     gnX400Address        = 3,
19218:     gnDirectoryName      = 4,
19219:     gnEDIPartyName       = 5,
19220:     gnUniformResourceIdentifier = 6,
19221:     gnIPAddress          = 7,
19222:     gnRegisteredID       = 8);
19223:
19224:   CL.AddTypeS('TX509GeneralNameType', '(gnOtherName, gnRFC822Name, gnDNSName, gnX400Address, gnDirectoryName,
19225:   gnEDIPartyName, gnUniformResourceIdentifier, gnIPAddress, gnRegisteredID)');
19226: Procedure InitX509Version( var A : TX509Version)');
19227: Function EncodeX509Version( const A : TX509Version) : AnsiString');
19228: Function DecodeX509Version( const TypeID : Byte; const Buf : string; const Size : Integer; var A : 
19229: TX509Version) : Integer');
19230: CL.AddTypeS('TX509Time', 'TDateTime');
19231: Function EncodeX509Time( const A : TX509Time) : AnsiString');
19232: Function DecodeX509Time(const TypeID : Byte; const Buf : string; const Size : Int; var A : TX509Time) : Integer;

```

```

19232: TX509Validity', 'record NotBefore:TX509Time; NotAfter : TX509Time; _Decoded: Boolean; end');
19233: // CL.AddTypeS('PX509Validity', '^TX509Validity // will not work');
19234: Function EncodeX509Validity( const A : TX509Validity) : AnsiString');
19235: Function DecodeX509Validity(const TypeID:Byte; const Buf : string; const Size : Integer; var A : TX509Validity) : Integer');
19236: CL.AddTypeS('TX509DHValidationParms','record Seed: AnsiString; PgenCounter : Integer; end');
19237: //CL.AddTypeS('PX509DHValidationParms', '^TX509DHValidationParms // will not work');
19238: CL.AddTypeS('TX509DHDomainParameters', 'record P : AnsiString; G : AnsiString';
19239: +' ; Q : AnsiString; J : AnsiString; ValidationParms : TX509DHValidationParms; end');
19240: // CL.AddTypeS('PX509DHDomainParameters', '^TX509DHDomainParameters // will not work');
19241: Function EncodeX509DHValidationParms( const A : TX509DHValidationParms) : AnsiString');
19242: Function DecodeX509DHValidationParms(const TypeID: Byte; const Buf:string; const Size: Integer; var A : TX509DHValidationParms): Integer');
19243: Function EncodeX509DHDomainParameters( const A : TX509DHDomainParameters) : AnsiString');
19244: Function DecodeX509DHDomainParameters(const TypeID: Byte;const Buf: string;const Size:Integer; var A : TX509DHDomainParameters) : Integer');
19245: CL.AddTypeS('TX509DSSParms', 'record P : AnsiString; Q : AnsiString; G : AnsiString; end');
19246: //CL.AddTypeS('PX509DSSParms', '^TX509DSSParms // will not work');
19247: Function EncodeX509DSSParms( const A : TX509DSSParms) : AnsiString');
19248: Function DecodeX509DSSParms(const TypeID: Byte; const Buf:string; const Size:Integer; var A : TX509DSSParms) : Integer');
19249: CL.AddTypeS('TX509DSSSigValue', 'record R : AnsiString; S : AnsiString; end');
19250: //CL.AddTypeS('PX509DSSSigValue', '^TX509DSSSigValue // will not work');
19251: Function EncodeX509DSSSigValue( const A : TX509DSSSigValue) : AnsiString');
19252: Function DecodeX509DSSSigValue( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509DSSSigValue) : Integer');
19253: CL.AddTypeS('TX509AlgorithmIdentifier','record Algorithm : TASN1ObjectIdentifier';
19254: +'fier; Parameters : AnsiString; _Decoded : Boolean; end');
19255: // CL.AddTypeS('PX509AlgorithmIdentifier', '^TX509AlgorithmIdentifier // will not work');
19256: Procedure InitX509AlgorithmIdentifier( var A : TX509AlgorithmIdentifier; const Algorithm : array of Integer; const Parameters : AnsiString)');
19257: Procedure InitX509AlgorithmIdentifierDSA(var A:TX509AlgorithmIdentifier;const Params:AnsiString);
19258: Function EncodeX509AlgorithmIdentifier( const A : TX509AlgorithmIdentifier) : AnsiString');
19259: Function DecodeX509AlgorithmIdentifier( const TypeID : Byte; const Buf : string; const Size : Integer;
var A : TX509AlgorithmIdentifier) : Integer');
19260: CL.AddTypeS('TX509RSAPublicKey','record Modulus:AnsiString;PublicExponent:AnsiString; end');
19261: //CL.AddTypeS('PX509RSAPublicKey', '^TX509RSAPublicKey // will not work');
19262: Function EncodeX509RSAPublicKey( const A : TX509RSAPublicKey) : AnsiString');
19263: Function DecodeX509RSAPublicKey( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509RSAPublicKey) : Integer');
19264: Function ParseX509RSAPublicKey(const Buf:string;const Size:Int;var A:TX509RSAPublicKey):Int;
19265: CL.AddTypeS('TX509DHPublicKey', 'AnsiString');
19266: Function EncodeX509DHPublicKey( const A : TX509DHPublicKey) : AnsiString');
19267: CL.AddTypeS('TX509DSAPublicKey', 'AnsiString');
19268: Function EncodeX509DSAPublicKey( const A : TX509DSAPublicKey) : AnsiString');
19269: CL.AddTypeS('TX509SubjectPublicKeyInfo', 'record Algorithm : TX509AlgorithmId';
19270: +'entifier; SubjectPublicKey : AnsiString; _Decoded : Boolean; end');
19271: //CL.AddTypeS('PX509SubjectPublicKeyInfo', '^TX509SubjectPublicKeyInfo // will not work');
19272: Procedure InitX509SubjectPublicKeyInfoDSA(var A : TX509SubjectPublicKeyInfo; const B : TX509DSSParms;
const PublicKey : AnsiString)');
19273: Function EncodeX509SubjectPublicKeyInfo( const A : TX509SubjectPublicKeyInfo) : AnsiString');
19274: Function DecodeX509SubjectPublicKeyInfo(const TypeID:Byte; const Buf:string; const Size:Integer; var A : TX509SubjectPublicKeyInfo) : Integer');
19275: Function ParseX509SubjectPublicKeyInfo( const Buf : string; const Size : Integer; var A : TX509SubjectPublicKeyInfo) : Integer');
19276: Function EncodeX509GeneralName(const A:TX509GeneralNameType;const EncodedName:AnsiString):AnsiString;
19277: Function DecodeX509GeneralName( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509GeneralNameType; var B : AnsiString) : Integer');
19278: TX509GeneralNames', 'array of record NameType:TX509GeneralNameType; Name : AnsiString; end');
19279: //CL.AddTypeS('PX509GeneralNames', '^TX509GeneralNames // will not work');
19280: Function EncodeX509GeneralNames( const A : TX509GeneralNames) : AnsiString');
19281: Function DecodeX509GeneralNames(const TypeID: Byte; const Buf : string; const Size : Integer; var A : TX509GeneralNames) : Integer');
19282: TX509BasicConstraints', 'record CA:Bool;PathLenConstraint:AnsiString;_DecodedCA:Boolean; end');
19283: //CL.AddTypeS('PX509BasicConstraints', '^TX509BasicConstraints // will not work');
19284: Function EncodeX509BasicConstraints( const A : TX509BasicConstraints) : AnsiString');
19285: Function DecodeX509BasicConstraints( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509BasicConstraints) : Integer');
19286: CL.AddTypeS(TX509AuthorityKeyIdentifier', 'record KeyIdentifier : AnsiString';
19287: +' ; AuthorityCertIssuer : TX509GeneralNames; AuthorityCertSerialNumber : Int64; end');
19288: //CL.AddTypeS('PX509AuthorityKeyIdentifier', '^TX509AuthorityKeyIdentifier //will not work');
19289: Function EncodeX509AuthorityKeyIdentifier(const A:TX509AuthorityKeyIdentifier): AnsiString');
19290: CL.AddTypeS('TX509SubjectKeyIdentifier', 'AnsiString');
19291: CL.AddTypeS(TX509KeyUsage', 'AnsiString');
19292: CL.AddTypeS(TX509Extension', 'record ExtnID : TASN1ObjectIdentifier; Critical';
19293: +'1 : Boolean; ExtnValue : AnsiString; _DecodedCritical : Boolean; _Decoded : Boolean; end');
19294: //CL.AddTypeS('PX509Extension', '^TX509Extension // will not work');
19295: Procedure InitX509ExtAuthorityKeyIdentifier( var A : TX509Extension; const B : TX509AuthorityKeyIdentifier)');
19296: Procedure InitX509ExtSubjectKeyIdentifier(var A:TX509Extension; const B:TX509SubjectKeyIdentifier);
19297: Procedure InitX509ExtBasicConstraints(var A:TX509Extension;const B:TX509BasicConstraints)');
19298: Function EncodeX509Extension( const A : TX509Extension) : AnsiString');
19299: Function DecodeX509Extension( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509Extension) : Integer');
19300: CL.AddTypeS('TX509Extensions', 'array of TX509Extension');
19301: //CL.AddTypeS('PX509Extensions', '^TX509Extensions // will not work');
19302: Procedure AppendX509Extensions( var A : TX509Extensions; const B : TX509Extension)');
19303: Function EncodeX509Extensions( const A : TX509Extensions) : AnsiString');
19304: Function DecodeX509Extensions( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509Extensions) : Integer');

```

```

19305: Function NormaliseX509IntKeyBuf( var KeyBuf : AnsiString) : Integer');
19306:   CL.AddTypeS('TX509TBSCertificate', 'record Version : TX509Version; SerialNumb'
19307:     +'er : AnsiString; Signature : TX509AlgorithmIdentifier; Issuer : TX509Name;' +
19308:     +' Validity : TX509Validity; Subject : TX509Name; SubjectPublicKeyInfo : TX5'
19309:     +'09SubjectPublicKeyInfo; IssuerUniqueID : AnsiString; SubjectUniqueID : Ans'
19310:     +'iString; Extensions: TX509Extensions; _DecodedVersion : Boolean; _Decoded : Boolean; end');
19311: // CL.AddTypeS('PX509TBSCertificate', '^TX509TBSCertificate // will not work');
19312: Function EncodeX509TBSCertificate( const A : TX509TBSCertificate) : AnsiString');
19313: Function DecodeX509TBSCertificate( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509TBSCertificate) : Integer';
19314:   CL.AddTypeS('TX509Certificate', 'record TBSCertificate : TX509TBSCertificate;' +
19315:     +' SignatureAlgorithm: TX509AlgorithmIdentifier;SignatureValue:AnsiString; _Decoded:Boolean;end');
19316: // CL.AddTypeS('PX509Certificate', '^TX509Certificate // will not work');
19317: CL.AddTypeS('TX509CertificateArray', 'array of TX509Certificate');
19318: Procedure InitX509Certificate( var A : TX509Certificate)';
19319: Function EncodeX509Certificate( const A : TX509Certificate) : AnsiString';
19320: Function DecodeX509Certificate(const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509Certificate) : Integer;
19321: Function ParseX509Certificate(const Buf: string; const Size:Integer; var A:TX509Certificate):Integer;
19322: Procedure ParseX509CertificateStr( const BufStr : AnsiString; var A : TX509Certificate)');
19323: Procedure ParseX509CertificatePEM( const BufStr : AnsiString; var A : TX509Certificate)');
19324: CL.AddTypeS('TX509RSAPrivateKey', 'record Version : Integer; Modulus : AnsiSt'
19325:   +'ring; PublicExponent : AnsiString; PrivateExponent : AnsiString; Prime1 : '
19326:   +'AnsiString; Prime2 : AnsiString; Exponent1 : AnsiString; Exponent2 : AnsiString; CRTCoefficient : AnsiString; _Decoded : Boolean; end');
19327: // CL.AddTypeS('PX509RSAPrivateKey', '^TX509RSAPrivateKey // will not work');
19328: Function EncodeX509RSAPrivateKey( const A : TX509RSAPrivateKey) : AnsiString';
19329: Function DecodeX509RSAPrivateKey( const TypeID : Byte; const Buf : string; const Size : Integer; var A : TX509RSAPrivateKey) : Integer';
19330: Function ParseX509RSAPrivateKey(const Buf:string:const Size:Integer;var A:TX509RSAPrivateKey):Integer;
19331: Procedure ParseX509RSAPrivateKeyStr(const BufStr : AnsiString; var A : TX509RSAPrivateKey)';
19332: Procedure ParseX509RSAPrivateKeyPEM(const BufStr : AnsiString; var A : TX509RSAPrivateKey)';
19333: Procedure SelfTestX509)';
19334: end;
19335:
19336: procedure SIRegister_FileStreamW(CL: TPSPPascalCompiler);
19337: begin
19338:   SIRegister_TFileStreamW(CL);
19339:   {TFormFadeSettings = record
19340:     Form: TForm;
19341:     Step: ShortInt;
19342:     DisableBlendOnFinish: Boolean;
19343:     Callback: TNotifyEvent;
19344:     MinAlpha, MaxAlpha: Byte;
19345:   end;}
19346:   TFormFadeSettings', 'record Form: TForm;Step: ShortInt; DisableBlendOnFinish: Boolean; Callback: TNotifyEvent; MinAlpha, MaxAlpha: Byte; end');
19347: CL.AddConstantN('fmForcePath','LongWord').SetUIInt( $80000000);
19348: Function LoadUnicodeFromStream( S : TStream; AsIsAnsi : Boolean) : WideString';
19349: function GetClipboardText: WideString;';
19350: procedure CopyToClipboard(Str: WideString);'
19351: function CurrentWinUser: WideString;';
19352:   function GetTempPathW: WideString;';
19353:   function GetTempFileNameW: WideString;';
19354:   function GetDesktopFolderW: WideString;';
19355:   function IsWritable(const FileName: WideString): Boolean;';
19356:   function SysErrorMessageW(ErrorCode: Integer): WideString;';
19357:   function FormatExceptionInfo: WideString;';
19358:   procedure ShowExceptionW(Message: WideString);'
19359:   procedure ChangeWindowStyle(const Form: HWND; Style: DWord; AddIt: Boolean);';
19360:   function SetNtfsCompressionW(const FileName: WideString; Level: Word): Boolean;';
19361:   function WriteWS(const Stream: TStream; const Str: WideString): Word;';
19362:   procedure FormFadeIn(Form: TForm; Step: ShortInt);'
19363:   procedure FormFadeOut(Form: TForm; Step: ShortInt);'
19364:   procedure FormFadeOutAndWait(Form: TForm; Step: ShortInt);'
19365:   function BrowseForFolderW(const Caption,DefaultPath: WideString;const OwnerWindow:HWND): WideString;
19366:   procedure FormFade(const Settings: TFormFadeSettings);'
19367:     function HashOfString(const Str: WideString): DWord;
19368:     function ComparePoints(const First, Second: TPoint): ShortInt;
19369:   end;
19370:
19371: procedure SIRegister_InetUtils(CL: TPSPPascalCompiler);
19372: begin
19373:   CL.AddTypeS(TNetUtilsSettings', 'record UserAgent : String; ProxyURL : Strin'
19374:     +'g; OpenURLFlags: DWord; TrafficCounter: Dword; UploadedCounter:DWord; ReadBufferSize:DWord;end');
19375:   TRawCharset', 'set of Char';
19376:   TInetHeaders', 'array of String';
19377:   SIRegister_EInet(CL);
19378:   TInetDownloadCallback', 'Function ( Downloaded, TotalSize : DWord) : Boolean';
19379:   Function InetDownloadTo( const DestFile : WideString; const URL : String; Callback : TInetDownloadCallback) : Boolean';
19380:   Function InetDownloadTo1( const DestFile : WideString; const URL : String; const Settings : TNetUtilsSettings; Callback : TInetDownloadCallback) : Boolean';
19381:   Function InetDownload(const URL:String;Dest:TStream;Callback: TInetDownloadCallback):Boolean;
19382:   Function InetDownload3( const URL : String; Dest : TStream; const Settings : TNetUtilsSettings; Callback : TInetDownloadCallback) : Boolean';
19383:   Function InetBufferedReadFrom( Handle : HInternet; Dest : TStream; const Settings:TNetUtilsSettings; Callback : TInetDownloadCallback) : DWord';
19384:   Function InetBufferedReadFrom5( Handle : HInternet; const Settings : TNetUtilsSettings; Callback : TInetDownloadCallback) : String';

```

```

19385: Function IsResponseStatusOK( Handle : HInternet ) : Boolean';
19386:   TMultipartItem', 'record Headers : TInetHeaders; Data : TStream; end');
19387: {   TMaskMatchInfo = record
19388:   Matched: Boolean;
19389:   StrPos: Word;
19390:   MatchLength: Word;
19391: end);
19392: TMultipartInfo', 'record Matched: Boolean; StrPos: Word; MatchLength: Word; end');
19393: TMultipartItems', 'array of TMultipartItem');
19394: TUploadFile', 'record Name: String; SourceFileName : WideString; Data : TStream; end');
19395: TUploadFiles', 'array of TUploadFile');
19396: Function FindBoundaryFor( const Items : TMultipartItems ) : String';
19397: Function RandomBoundary : String';
19398: Function GenerateMultipartFormFrom(const Items:TMultipartItems;out ExtraHeaders:TInetHeaders):String;
19399: Function InetUploadTo( const ToURL : String; const Headers : TInetHeaders; const Items : TMultipartItems;
const Settings : TNetUtilsSettings) : String;');
19400: Function InetUploadTo7(const ToURL: String; const Items: TMultipartItems; const Settings :
TNetUtilsSettings : String;');
19401: Function InetUploadTo8(const ToURL: String; const Items: TMultipartItems) : String;');
19402: Function InetUploadStreamsTo(const ToURL: String; const Settings:TNetUtilsSettings; Streams :
TUploadFiles) : String;');
19403: Function InetUploadStreamsTo10(const ToURL:String;const Streams:TUploadFiles):String;
19404: Function InetUploadFileTo( const ToURL : String; const Settings : TNetUtilsSettings; const ItemName : String;
const FilePath : WideString) : String;');
19405: Function InetUploadFileTo12(const ToURL:String;const ItemName:String;const FilePath:WideString) : String;
19406: Function InetUploadFilesTo( const ToURL : String; const Settings : TNetUtilsSettings; const Files : array
of const) : String;');
19407: Function InetUploadFilesTo14(const ToURL: String;const Files: array of const):String;
19408: Function AppendQueryTo(const URL: String; const Arguments: array of const):String');
19409: Function HasQueryPart( const URL : String) : Boolean';
19410: Function BuildQueryFrom( const Arguments : array of const) : String';
19411: Function BuildURLW(Protocol,Host:String;Port:Word;Path,Script:String;const Arguments:array of const)
:String;
19412: Function CustomEncode(const Str: WideString; const RawChars:TRawCharset) : String';
19413: Function EncodeURI( const Str : WideString) : String';
19414: Function EncodeURIComponent( const Str : WideString) : String';
19415: Procedure InetGetLastError( out ErrorCode : DWord; out ErrorMessage : String );
19416: Function InetGetLastErrorCode : DWord';
19417: Function InetGetLastErrorMsg : String';
19418: Function AbsoluteURLFrom( URL, BaseURL, basePath : String ) : String';
19419: Procedure SplitURL( const URL : String; out Domain, Path : String );
19420: Function DomainOf( const URL : String ) : String';
19421: Function PathFromURL( const URL : String ) : String';
19422: Function InetHeaders( const NameValues : array of const ) : TInetHeaders';
19423: Function NoInetHeaders : TInetHeaders');
19424: Function JoinHeaders( const Headers : TInetHeaders ) : String';
19425: CL.AddConstantN('InetHeaderEOLN','String').SetString( #13#10 );
19426: Procedure SetDefaultNetUtilsSettings();
19427: Function TotalDownTrafficThroughNetUtils : DWord';
19428: TDBDraw', 'record DisplayDC : HDC; MemDC : HDC; MemBitmap : HBIT'
+ 'MAP; OldBitmap : HBITMAP; OldFont : HFONT; OldPen : HPEN; end');
19429: TPieceFormatData', 'record Position : TMaskMatchInfo; Color : TColor; end');
19430: TFormatData', 'array of TPieceFormatData');
19431: TDrawFormattedTextSettings', 'record Text : WideString; FormatDa
+ 'ta : TFormatData; Canvas : TCanvas; WrapText : Boolean; DestPos : TPoint;
19432: +'MaxWidth : Word; CharSpacing : Word; end');
19433: TWrapTextSettings', 'record DC : HDC; Str : WideString; Delimit'
+ 'r : WideString; MaxWidth:Word; LeftMargin:Word; CharSpacing:Word;LastChar : TSize; end');
19434: Function TextSize( const DC : HDC; const Str : WideString) : TSize';
19435: TextWidthW2( const DC : HDC; const Str : WideString) : Integer';
19436: TextHeightW2( const DC : HDC; const Str : WideString) : Integer';
19437: GetLineHeightOf( const Font : HFONT ) : Word';
19438: TextWidthEx( const DC:HDC,const Str:WideString; const CharSpacing:Word): Integer;
19439: TextHeightEx( const DC:HDC,const Str:WideString; const CharSpacing:Word): Integer;
19440: TextSizeEx( const DC:HDC,const Str:WideString; const CharSpacing:Word):TSize';
19441: TextWithBreaksSize( Settings : TWrapTextSettings ) : TSize;';
19442: DoubleBufferedDraw( const DisplaySurface:HDC;const BufferSize:TPoint):TDBDraw;
19443: DoubleBufferedDraw17( const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
19444: DoubleBufferedDraw2( const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
19445: Procedure DrawFormattedText( const Settings : TDrawFormattedTextSettings );
19446: Function GetLastCharPos(const DC:HDC;const Str:WideString;const MaxWidth:Word;const ChrSpacing:Word)
:TSize;
19447: WrapNonMonospacedText( const DC :HDC; const Str : WideString; const Delimiter:WideString; const
MaxWidth : Word; const CharSpacing : Word ) : WideString;');
19448: Function WrapNonMonospacedText2( var Settings : TWrapTextSettings ) : WideString;');
19449: end;
19450: procedure SIRegister_StrConv(CL: TPPascalCompiler);
19451: begin
19452:   CL.AddTypeS('TCodepage', 'DWord');
19453:   CL.AddConstantN('CP_INVALIDID','LongInt').SetInt( TCodepage ( - 1 ) );
19454:   CL.AddConstantN('CP_ASIS','LongInt').SetInt( TCodepage ( - 2 ) );
19455:   CL.AddConstantN('CP_ANSI','longint').SetInt(0);
19456:   CL.AddConstantN('CP_OEM','longint').SetInt(1);
19457:   CL.AddConstantN('CP_SHIFTJIS','LongInt').SetInt( 932 );
19458:   CL.AddConstantN('CP_LATIN1','LongInt').SetInt( 1250 );
19459:   CL.AddConstantN('CP_UNICODE','LongInt').SetInt( 1200 );
19460:   CL.AddConstantN('CP_UTF8','LongInt').SetInt( 65001 );
19461:   Function MinStrConvBufSize( SrcCodepage : TCodepage; Str : String ) : Integer;');

```

```

19466: Function MinStrConvBufSize1( DestCodepage : TCodepage; Wide : WideString) : Integer;';
19467: Function ToWideString(SrcCodepage: TCodepage; Str : String; BufSize: Integer) : WideString');
19468: Function FromWideString(DestCodepage:TCodepage;Str:WideString;BufSize:Integer;Fail:Boolean):String;
19469: Function CharsetToID( Str : String) : TCodepage';
19470: Function IdToCharset( ID : TCodepage; GetDescription : Boolean) : String');
19471: function CompareStrW(const S1, S2: WideString; Flags: DWord = 0): Integer;
19472: function CompareTextW(const S1, S2: WideString): Integer;
19473: function MaskMatch(const Str, Mask: WideString): Boolean;
19474: { Info can have special values in some cases:
19475:   * Matched = True but MatchLength = 0 (and StrPos having random value) - this means that Mask consisted
19476:     of only "*" and
19477:       thus no particular substring could be specified (since it could match any part of the string). }
19478: function MaskMatchInfo(const Str,Mask:WideString;StartingPos:Word = 1):TMaskMatchInfo;
19479: //Strutils -----
19479: Function TryStrToIntStrict(const S: String; out Value:Integer; Min:Integer):Boolean;
19480: Function TryStrToFloatStrict( const S : String; out Value : Single; const FormatSettings : TFormatSettings ) : Boolean;';
19481: Function TryStrToFloatStrict1( const S : String; out Value : Double; const FormatSettings : TFormatSettings ) : Boolean;';
19482: Function DetectEolnStyleIn( const Str : WideString) : WideString';
19483: Function DetectEolnStyleInANSI( Stream : TStream) : WideString';
19484: Function PascalQuote( const Str : WideString) : WideString';
19485: Function StrRepeatW( const Str : WideString; Times : Integer) : WideString';
19486: Function EscapeString(const Str: WideString; CharsToEscape:WideString): WideString';
19487: Function UnescapeString(const Str: WideString; CharsToEscape: WideString):WideString;
19488: Function BinToHexW( const Buf : String; Delim : String) : String';
19489: Function HexToBinW( Text : String) : String';
19490: Function SoftHexToBin( Text : String) : String';
19491: Function FormatVersion( Version : Word) : WideString';
19492: Function FormatDateW( Date : DWord) : WideString';
19493: Function FormatNumber( Number : DWord) : WideString';
19494: Function GenericFormat(Number:Single;const Language:TGenericFormatLanguage):WideString;
19495: Function FormatInterval( Millisecs : DWord) : WideString';
19496: Function FormatSize( Bytes : DWord) : WideString';
19497: Function PosLast( const Substr, Str : String; Start : Word) : Integer';
19498: Function PosLastW( const Substr, Str : WideString; Start : Word) : Integer';
19499: Function IsDelimiterW(const Delimiters,S: WideString; Index: Integer): Boolean';
19500: Function RemoveNonWordChars(const Str: WideString;DoNotRemove:WideString):WideString;
19501: Function IsQuoteChar( const aChr : Char) : Boolean';
19502: Function WrapTextW(const Str:WideString;const Delimiter:WideString;const MaxWidth:Word):WideString;
19503: Function PadText(const Str:WideString;const NewLine:WideString;const MaxWidth:Word):WideString;
19504: Function PadTextWithVariableLineLength(const Str : WideString; const NewLine, PadStr : WideString; const LineLengths : array of Integer) : WideString';
19505: Function StrPadW(const Str:WideString; ToLength:Integer; PadChar: WideChar) : WideString';
19506: Function StrPadLeftW(const Str:WideString;ToLength:Integer;PadChar: WideChar): WideString';
19507: //Function StrRepeat( const Str : WideString; Times : Integer) : WideString';
19508: Function StrReverseW( const Str : WideString) : WideString';
19509: Function CountSubstr( const Substr, Str : WideString) : Integer';
19510: Procedure DeleteArrayItem( var A : TWideStringArray; Index : Integer );
19511: Function TrimStringArray( WSArray : TWideStringArray ) : TWideStringArray';
19512: Function TrimWS( Str : WideString; const Chars : WideString) : WideString';
19513: Function TrimLeftWS( Str : WideString; const Chars : WideString) : WideString';
19514: //Function TrimRightWS( Str : WideString; const Chars : WideString) : WideString';
19515: Function ConsistsOfChars( const Str, Chars : WideString) : Boolean';
19516: Function UpperCaseW( const Str : WideString) : WideString';
19517: Function LowerCaseW( const Str : WideString) : WideString';
19518: Function UpperCaseFirst( const Str : WideString) : WideString';
19519: Function LowerCaseFirst( const Str : WideString) : WideString';
19520: Function StripAccelChars( const Str : WideString) : WideString';
19521: end;
19522:
19523: procedure SIRegister_REXX(CL: TPPascalCompiler);
19524: begin
19525: CL.AddTypeS('TState', '( TrimLeader, StartToken, EndToken )');
19526: CL.AddTypeS('TStrIndex', 'LongInt');
19527: CL.AddTypeS('TTokIndex', 'WORD');
19528: CL.AddTypeS('TStrIndexB', 'BYTE');
19529: CL.AddTypeS('TTokIndexB', 'BYTE');
19530: CL.AddClassN(CL.FindClass('TOBJECT'), 'EConversionError');
19531: Function Abbrev(const information, info: STRING; const nMatch: TStrIndex) : BOOLEAN';
19532: Function AllSame( const s : STRING; const c : CHAR) : BOOLEAN';
19533: Function Capitalize( const s : STRING) : STRING';
19534: Function Center( const s : STRING; const sLength : TStrIndex) : STRING';
19535: Function Left( const s : STRING; const sLength : TStrIndex) : STRING';
19536: Function Right( const s : STRING; const sLength : TStrIndex) : STRING';
19537: Function Copies( const s : STRING; const n : TStrIndex) : STRING';
19538: Function CountChar( const s : STRING; const c : CHAR) : TStrIndex';
19539: Function DeleteStringREXX( const substring : STRING; const s : STRING) : STRING';
19540: Function Overlay( const ovly, target : STRING; const n : TStrIndex) : STRING';
19541: Function Plural( const n: LongInt; const singularform, pluralform : STRING) : STRING';
19542: Function Reverse( const s : STRING) : STRING';
19543: Function Spacerexx( const s : STRING; const n : TStrIndex) : STRING';
19544: Function StripREXX( const s : STRING; const option : STRING) : STRING';
19545: Function TestString( const sLength : TStrIndex) : STRING';
19546: Function Translate( const s, OutTable, InTable : STRING) : STRING';
19547: Function XRange( const start, stop : BYTE) : STRING';
19548: Function B2X( const b : BYTE) : STRING';
19549: Function C2D( const s : STRING) : DOUBLE';
19550: Function C2I( const s : STRING) : INTEGER';

```

```

19551: Function C2L( const s : STRING ) : LONGINT';
19552: Function C2W( const s : STRING ) : WORD';
19553: Function C2X( const s : STRING ) : STRING';
19554: Function I2C( const i : INTEGER ) : STRING';
19555: Function I2X( const i : INTEGER ) : STRING';
19556: Function L2C( const i : LONGINT ) : STRING';
19557: Function L2X( const i : LONGINT ) : STRING';
19558: Function D2C( const x : DOUBLE; const d : BYTE ) : STRING';
19559: Function W2C( const w : WORD ) : STRING';
19560: Function W2X( const w : WORD ) : STRING';
19561: Function X2W( const s : STRING ) : WORD';
19562: Function JulianDate( const DateTime : TDateTime ) : LongInt';
19563: Function TimeDifference( const StartTime, StopTime : TDateTime ) : DOUBLE';
19564: Function Pwr( const x, y : DOUBLE ) : DOUBLE';
19565: end;
19566:
19567: procedure SIRegister_StringGridLibrary(CL: TPSPascalCompiler);
19568: begin
19569: Procedure ReadGridFile( var StringGrid : TStringGrid; Gridfile : STRING' );
19570: Procedure WriteGridFile( var StringGrid : TStringGrid; Gridfile : STRING' );
19571: Procedure AddBlankRowToTop( var StringGrid : TStringGrid' );
19572: Procedure DeleteSelectedRow( var StringGrid : TStringGrid' );
19573: Function StringGridSearch(const StringGrid : TStringGrid; const column : INTEGER; const target : STRING
: INTEGER');
19574: Function XLeft( rect : TRect; canvas : TCanvas; s : STRING ) : INTEGER';
19575: Function XCenter( rect : TRect; canvas : TCanvas; s : STRING ) : INTEGER';
19576: Function XRight( rect : TRect; canvas : TCanvas; s : STRING ) : INTEGER';
19577: Function YCenter( rect : TRect; canvas : TCanvas; s : STRING ) : INTEGER';
19578: end;
19579:
19580: procedure SIRegister_InetUtils2(CL: TPSPascalCompiler);
19581: begin
19582: Procedure AdjustArray(var DWArray:array of DWord;const Delta: Integer; MinValueToAdjust:DWord);
19583: Function GetDroppedFileNames( const DropID : Integer ) : TWideStringArray';
19584: Function AreBytesEqual( const First, Second : array of Byte ) : Boolean';
19585: Function AreBytesEqual( const First, Second, Length : DWord ) : Boolean';
19586: Function MaskForBytes( const NumberOfBytes : Byte ) : DWord';
19587: Function IntToBinByte( Int : Byte ) : String';
19588: Function IntToBinWord( Int : Word ) : String';
19589: Function IntToBinDWord( Int : DWord; Digits : Byte; SpaceEach : Byte ) : String';
19590: Function WriteWS( const Stream : TStream; const Str : WideString ) : Word';
19591: Procedure WriteArray( const Stream : TStream; const WSArray : array of WideString );'
19592: Procedure WriteArray6( const Stream : TStream; const DWArray : array of DWord );'
19593: Function ReadWS( const Stream : TStream ) : WideString';
19594: Function ReadWS8( const Stream : TStream; out Len : Word ) : WideString';
19595: Procedure ReadArray( const Stream : TStream; var WSArray : array of WideString );'
19596: Procedure ReadArray10( const Stream : TStream; var DWArray : array of DWord );'
19597: Function ParamStrW( Index : Integer ) : WideString';
19598: Function ParamStrFrom( CmdLine : WideString; Index : Integer ) : WideString';
19599: Function ParamStrEx( Cmdline : WideString; Index : Integer; out Pos : Integer ) : WideString';
19600: Procedure FindMask( Mask : WideString; Result : TStringsW );
19601: Procedure FindAll( BasePath, Mask : WideString; Result : TStringsW );
19602: Procedure FindAllRelative( BasePath, Mask : WideString; Result : TStringsW );
19603: Function IsValidPathChar( const Char : WideChar ) : Boolean';
19604: Function MakeValidFileNameW( const Str:WideString;const SubstitutionChar:WideChar ):WideString;
19605: Function ExtractFilePathW( FileName : WideString ) : WideString';
19606: Function ExtractFileNameW( Path : WideString ) : WideString';
19607: Function ExpandFileNameW( FileName : WideString ) : WideString';
19608: Function ExpandFileName12( FileName, basePath : WideString ) : WideString';
19609: Function CurrentDirectory : WideString';
19610: Function ChDirW( const ToPath : WideString ) : Boolean';
19611: Function ExtractFileExtW( FileName : WideString ) : WideString';
19612: Function ChangeFileExtW2( FileName, Extension : WideString ) : WideString';
19613: Function IncludeTrailingBackslashW( Path : WideString ) : WideString';
19614: Function ExcludeTrailingBackslashW( Path : WideString ) : WideString';
19615: Function IncludeTrailingPathDelimiterW( Path : WideString ) : WideString';
19616: Function ExcludeTrailingPathDelimiterW( Path : WideString ) : WideString';
19617: Function IncludeLeadingPathDelimiter( Path : WideString ) : WideString';
19618: Function ExcludeLeadingPathDelimiter( Path : WideString ) : WideString';
19619: Function FileInfo( Path : WideString ) : TWin32FindData';
19620: Function IsDirectoryW( Path : WideString ) : Boolean';
19621: Function FileAgeW( const FileName : WideString ) : Integer';
19622: Function FileExistsW( Path : WideString ) : Boolean';
19623: Function FilesizeW( Path : WideString ) : DWord';
19624: Function FileSize64( Path : WideString ) : Int64';
19625: Function DeleteFileW( Path : WideString ) : Boolean';
19626: Function CopyDirectoryW( Source, Destination : WideString ) : Boolean';
19627: Function RemoveDirectoryW( Path : WideString ) : Boolean';
19628: Function ForceDirectoriesW( Path : WideString ) : Boolean';
19629: Function MkDirW( Path : WideString ) : Boolean';
19630: Function GetEnvironmentVariableW( Name : WideString ) : WideString';
19631: Function ResolveEnvVars(Path:WideString; Callback:TEnvVarResolver;Unescape:Boolean):WideString;
19632: Function ResolveEnvVars14( Path : WideString; Unescape : Boolean ) : WideString';
19633: Function ReadRegValue( Root : DWord; const Path, Key : WideString ) : WideString';
19634: Function FadeSettings(Form:TForm; Step: ShortInt; Callback:TNotifyEvent): TFormFadeSettings;';
19635: Function FadeSettings17(Form:TForm; MinAlpha,MaxAlpha:Byte; Step:ShortInt): TFormFadeSettings;;
19636: Function GetSpecialFolderID( const Path : String ) : Integer';
19637: Function GetSpecialFolderPath2( FolderID : Integer ) : String';
19638: end;

```

```

19639:
19640: procedure SIRegister_KFunctions(CL: TPSPascalCompiler);
19641: begin
19642:   SHFolderDll', 'String').SetString( 'SHFolder.dll');
19643:   CL.AddConstantN('KM_BASE','LongInt').SetInt( LM_USER + 1024);
19644:   CL.AddConstantN('KM_LATEUPDATE','LongInt').SetInt( KM_BASE + 1);
19645:   crHResize','LongInt').SetInt( TCursor ( 101 ));
19646:   CL.AddConstantN('crVResize','LongInt').SetInt( TCursor ( 102 ));
19647:   crDragHandFree','LongInt').SetInt( TCursor ( 103 ));
19648:   crDragHandGrip','LongInt').SetInt( TCursor ( 104 ));
19649:   cCheckBoxFrameSize','LongInt').SetInt( 13);
19650:   cCR','Char').SetString( #13);
19651:   cLF','Char').SetString( #10);
19652:   cTAB','Char').SetString( #9);
19653:   cSPACE','Char').SetString( #32);
19654:   cNULL','Char').SetString( #0);
19655:   {cWordBreaks','LongInt').Value.ts32 := ord(cNULL) or ord(cTAB) or ord(cSPACE);
19656:   cLineBreaks','LongInt').Value.ts32 := ord(cCR) or ord(cLF);
19657:   cEllipsis','String').SetString( '... ');
19658:   cEOL','').SetString( cLF);
19659:   cFirstEOL','').SetString( cLF);
19660:   cFirstEOL','').SetString( cCR); }
19661:   CL.AddTypeS('TLMMessage', 'TMessage');
19662:   CL.AddTypeS('TKkString', 'WideString');
19663:   CL.AddTypeS('TKkChar', 'char');
19664: //TKChar = UTF8Char;
19665:   CL.AddTypeS('LONG_PTR2', 'Longint');
19666:   CL.AddTypeS('TKSysCharSet', 'set of AnsiChar');
19667:   CL.AddTypeS('TKCurrencyFormat', 'record CurrencyFormat : Byte; CurrencyDecimal'
19668:     +'s : Byte; CurrencyString : TKkString; DecimalSep : Char; ThousandSep : Cha'
19669:     +'r; UseThousandsSep : Boolean; end');
19670: {CL.AddTypeS('TKAppContext', 'record Application : TApplication; Screen : TScr'
19671:   +'een; GlobalNameSpace : IReadWriteSync; MainThreadID : LongWord; IntConstLi'
19672:   +'st : TThreadList; WidgetSet : TWidgetSet; DragManager : TDragManager; end'); }
19673: // CL.AddTypeS('PKAppContext', '^TKAppContext // will not work');
19674: //CL.AddTypeS('TKClipboardFormat', 'TClipboardFormat');
19675: CL.AddTypeS('TKClipboardFormat', 'Word');
19676: CL.AddTypeS('TKCellSpan', 'record ColSpan : Integer; RowSpan : Integer; end');
19677: CL.AddDelphiFunction('Function AdjustDecimalSeparator( const S : string ) : string');
19678: Function AnsiStringToString( const Text : AnsiString; CodePage : Cardinal ) : TKkString;
19679: //Function BinarySearch( AData : string; ACount : Integer; KeyPtr : string; ACompareProc :
19680:   TBSSCompareProc; ASortedDown : Boolean ) : Integer';
19681: Procedure CallTrackMouseEvent( Control : TWInControl; var Status : Boolean );
19682: Procedure CenterWindowInWindow( CenteredWnd, BoundWnd : HWnd );
19683: Procedure CenterWindowOnScreen( CenteredWnd : HWnd );
19684: Function CharInSetEx( AChar : AnsiChar; const ASet : TKSysCharSet ) : Boolean';
19685: Function CharInSetEx1( AChar : WideChar; const ASet : TKSysCharSet ) : Boolean';
19686: Function ClipboardLoadStreamAs( const AFormat:string;ASTream:TStream;var AText:TKkString ):Bool;
19687: Function ClipboardSaveStreamAs( const AFormat:string;ASStream:TStream;const AText:TKkString ):Bool;
19688: Function CompareIntegers( I1, I2 : Integer ) : Integer';
19689: //Function CompareWideChars( W1, W2 : PWideChar; Locale : Cardinal ) : Integer';
19690: // Function CompareChars( S1, S2 : PChar; Locale : Cardinal ) : Integer';
19691: // Function CompareWideStrings( W1, W2 : WideString; Locale : Cardinal ) : Integer';
19692: Procedure ConvertTabsToSpaces( var AText : TKkString; ASpacesForTab : Integer );
19693: Function CreateMultipleDir( const Dir : string ) : Boolean';
19694: Function DigitToNibble( Digit : AnsiChar; var Nibble : Byte ) : Boolean';
19695: Function DivUp( Dividend, Divisor : Integer ) : Integer';
19696: Function DivDown( Dividend, Divisor : Integer ) : Integer';
19697: Function EditIsFocused( AMustAllowWrite : Boolean ) : Boolean';
19698: Function EditFocusedTextCanCopy : Boolean';
19699: Function EditFocusedTextCanCut : Boolean';
19700: Function EditFocusedTextCanDelete : Boolean';
19701: Function EditFocusedTextCanPaste : Boolean';
19702: Function EditFocusedTextCanUndo : Boolean';
19703: Procedure EditUndoFocused';
19704: Procedure EditDeleteFocused';
19705: Procedure EditCutFocused';
19706: Procedure EditCopyFocused';
19707: Procedure EditPasteFocused';
19708: Procedure EditSelectAllFocused';
19709: Procedure EnableControls2( AParent : TWInControl; AEnabled, ARecursive : Boolean );
19710: Procedure EnsureLastPathSlash( var APath : string );
19711: Procedure Error2( const Msg : string );
19712: Function FillMessage( Msg : Cardinal; WParam : WPARAM; LParam : LPARAM ) : TLMMessage';
19713: Function FormatCurrency( Value : Currency; const AFormat : TKCurrencyFormat ) : TKkString';
19714: //Function GetAppContext( var Ctx : TKAppContext ) : Boolean';
19715: Function GetAppVersion( const ALibName : string; var MajorVersion, MinorVersion, BuildNumber,
19716:   RevisionNumber : Word ) : Boolean';
19717: Function GetCharCount( const AText : TKkString; AChar : TKkChar ) : Integer';
19718: Function GetControlText( Value : TWInControl ) : TKkString';
19719: Function GetFormatSettings2 : TFormatSettings';
19720: Function IntToAscii( Value : Int64; Digits : Integer ) : string';
19721: Function IntToBinStr( Value : Int64; Digits : Byte; const Suffix : string ) : string';
19722: Function IntToBCD( Value : Cardinal ) : Cardinal';
19723: Function IntToDecStr( Value : Int64 ) : string';
19724: Function IntToHexStr( Value : Int64; Digits : Byte; const Prefix, Suffix : string; UseLowerCase :
19725:   Boolean ) : string';

```

```

19725: Function IntToOctStr( Value : Int64 ) : string';
19726: Function IntToRoman2( Value : Integer; AUpperCase : Boolean) : string');
19727: Function IntToLatin( Value : Integer; AUpperCase : Boolean) : string');
19728: Function IntPowerInt( Value : Int64; Exponent : Integer) : Int64');
19729: Function AsciiToInt( S : string; Digits : Integer) : Int64');
19730: Function BCDToInt( Value : Cardinal) : Cardinal');
19731: Function BinStrToInt2(S:string; Digits:Byte; Signed : Boolean; var Code : Integer) : Int64');
19732: Function DecStrToInt( S : string; var Code : Integer) : Int64');
19733: Function HexStrToInt(S : string; Digits: Byte; Signed:Boolean; var Code : Integer) : Int64');
19734: Function OctStrToInt( S : string; var Code : Integer) : Int64');
19735: Function KFormat14( const Format : string; const Args : array of const; const AFormatSettings : TFormatSettings) : string');
19736: Function KFormat15( const Format : WideString; const Args : array of const; const AFormatSettings : TFormatSettings) : WideString');
19737: Function MakeCellSpan( AColumns, ARows : Integer) : TKCellSpan');
19738: Function MinMax16( Value, Min, Max : ShortInt) : ShortInt');
19739: Function MinMax17( Value, Min, Max : SmallInt) : SmallInt');
19740: Function MinMax18( Value, Min, Max : Integer) : Integer');
19741: Function MinMax19( Value, Min, Max : Int64) : Int64');
19742: Function MinMax20( Value, Min, Max : Single) : Single');
19743: Function MinMax21( Value, Min, Max : Double) : Double');
19744: Function MinMax22( Value, Min, Max : Extended) : Extended');
19745: Function NibbleToDigit( Nibble : Byte; UpperCase : Boolean) : AnsiChar');
19746: Procedure OpenURLWithShell( const AText : TKkString)');
19747: //Procedure QuickSortNR( AData : Pointer; ACount : Integer; ACompareProc : TQsCompareProc; AExchangeProc : TQsExchangeProc; ASortedDown : Boolean)');
19748: //Procedure QuickSort( AData : Pointer; ACount : Integer; ACompareProc : TQsCompareProc; AExchangeProc : TQsExchangeProc; ASortedDown : Boolean)');
19749: Procedure OffsetPoint23( var APoint : TPoint; AX, AY : Integer);');
19750: Procedure OffsetPoint24( var APoint : TPoint; const AOffset : TPoint);');
19751: Function RectInRect( Bounds, Rect : TRect) : Boolean');
19752: Procedure OffsetRect25( var ARect : TRect; AX, AY : Integer);');
19753: Procedure OffsetRect26( var ARect : TRect; const AOffset : TPoint);');
19754: //Function SetAppContext( const Ctx : TKAppContext) : Boolean');
19755: Procedure SetControlClipRect( AControl : TWinControl; const ARect : TRect)');
19756: Procedure SetControlText( Value : TWinControl; const Text : TKkString)');
19757: Procedure StripLastPathSlash( var APath : string)');
19758: Function StrNextCharIndex( const AText : TKkString; Index : Integer) : Integer');
19759: Function StrPreviousCharIndex( const AText : TKkString; Index : Integer) : Integer');
19760: Function StringCharBegin( const AText : TKkString; Index : Integer) : Integer');
19761: Function StringLength( const AText : TKkString) : Integer');
19762: Function StringCopy( const ASource : TKkString; At, Count : Integer) : TKkString');
19763: Procedure StringDelete( var ASource : TKkString; At, Count : Integer)');
19764: Procedure TrimWhiteSpaces27(const AText:TKkString; var AStart,ALen:Integer;const ASet:TKSysCharSet);
19765: Procedure TrimWhiteSpaces28( var AText : TKkString; const ASet : TKSysCharSet);');
19766: Procedure TrimWhiteSpaces29( var AText : AnsiString; const ASet : TKSysCharSet);');
19767: Function StringToAnsiString( const AText : TKkString; CodePage : Cardinal) : AnsiString');
19768: Function StringToChar( const AText : TKkString; AIndex : Integer) : TKkChar');
19769: Function GetWindowsFolder2( CSIDL : Cardinal; var APath : string) : Boolean');
19770: Function RunExecutable( const AFileName : string; AWaitForIt : Boolean) : DWORD');
19771: Function SystemCodePage : Integer');
19772: Function NativeUTFToUnicode( const AText : TKkString) : WideString');
19773: Function UnicodeUpperCase2( const AText : TKkString) : TKkString');
19774: Function UnicodeLowerCase2( const AText : TKkString) : TKkString');
19775: Function UnicodeToNativeUTF( const AParam : WideChar) : TKkString');
19776: Function UnicodeStringReplace( const AText, AOldPattern, ANewPattern : TKkString; AFLags : TReplaceFlags) : TKkString');
19777: end;
19778:
19779: procedure SIRegister_KMessageBox(CL: TPSPascalCompiler);
19780: begin
19781:   CL.AddTypeS('TKMsgBoxButton', '( mbYes, mbNo, mbOK, mbCancel, mbClose, mbAbort' +
19782:     +'t, mbRetry, mbIgnore, mbAll, mbNoToAll, mbYesToAll, mbHelp )');
19783:   CL.AddTypeS('TKMsgBoxIcon', '( miNone, miInformation, miQuestion, miWarning, miStop )');
19784:   Function CreateMsgBox( const Caption, Text : string; const Buttons : array of TKMsgBoxButton; Icon : TKMsgBoxIcon; Def : integer) : TCustomForm');
19785:   Function CreateMsgBoxEx( const Caption, Text : string; const BtNs : array of string; Icon : TKMsgBoxIcon; Def : integer) : TCustomForm');
19786:   Procedure FreeMsgBox( AMsgBox : TCustomForm');
19787:   Function KMMsgBox( const Caption, Text : string; const Buttons : array of TKMsgBoxButton; Icon : TKMsgBoxIcon; Def : integer) : integer');
19788:   Function KMMsgBoxEx(const Caption, Text: string; const Buttons : array of string; Icon : TKMsgBoxIcon; Def : integer) : integer');
19789:   Function KInputBox( const Caption, Prompt : string; var Text : string) : TModalResult';
19790:   //Function KNumberInputBox( const ACaption, APrompt : string; var AValue : double; AMin, AMax : double; AFormats : TKNumericUpDownAcceptedFormats) : TModalResult';
19791:   TKMsgBoxButtons', '( mbAbortRetryIgnore, mbOkOnly, mbOkCancel,mbRetryCancel,mbYesNo,mbYesNoCancel)');
19792:   Function MsgBox2(const Caption,Text:string;const Buttons:TKMsgBoxButtons;Icon:TKMsgBoxIcon):integer;
19793:   Function AppMsgBox(const Caption, Text : string; Flags : integer) : integer');
19794: end;
19795:
19796: procedure SIRegister_Kronos(CL: TPSPascalCompiler);
19797: begin
19798:   'ChurchDayCount','LongInt').SetInt( 21 );
19799:   CL.AddConstantN('CommonDayCount','LongInt').SetInt( 4 );
19800:   'chAdvent1','LongInt').SetInt( 1 );
19801:   'chAdvent2','LongInt').SetInt( 2 );
19802:   'chAdvent3','LongInt').SetInt( 3 );
19803:   'chAdvent4','LongInt').SetInt( 4 );

```

```

19804: 'chChristmasEve','LongInt').SetInt( 5);
19805: 'chChristmasDay','LongInt').SetInt( 6);
19806: 'chBoxingDay','LongInt').SetInt( 7);
19807: 'chNewYearEve','LongInt').SetInt( 8);
19808: 'chNewYearDay','LongInt').SetInt( 9);
19809: 'chShroveTuesday','LongInt').SetInt( 10);
19810: 'chAshWednesday','LongInt').SetInt( 11);
19811: 'chPalmSunday','LongInt').SetInt( 12);
19812: 'chMaundyThursday','LongInt').SetInt( 13);
19813: 'chGoodFriday','LongInt').SetInt( 14);
19814: 'chEasterEve','LongInt').SetInt( 15);
19815: 'chEasterSunday','LongInt').SetInt( 16);
19816: 'chEasterMonday','LongInt').SetInt( 17);
19817: 'chWhitEve','LongInt').SetInt( 18);
19818: 'chWhitSunday','LongInt').SetInt( 19);
19819: 'chWhitMonday','LongInt').SetInt( 20);
19820: 'chAscensionDay','LongInt').SetInt( 21);
19821: 'coUNDay','LongInt').SetInt( 22);
19822: 'coWomensDay','LongInt').SetInt( 23);
19823: 'coMayDay','LongInt').SetInt( 24);
19824: 'coLiteracyDay','LongInt').SetInt( 25);
19825: 'UserDayType','LongInt').SetInt( 26);
19826: //UserDayType = ChurchDayCount + CommonDayCount + 1;
19827: TFIRSTLASTNUMBER, 'array[1..2] of word;');
19828: TDaytypeID, 'array[1..255] of word;');
19829: TMonthImage2, 'array[0..7] of smallint;');
19830: TMonthImage, 'array[1..6] of tmonthimage2;');
19831: //TMonthImage = array[1..6, 0..7] of smallint;
19832: //TFIRSTLASTNUMBER = array[1..2] of word;
19833: //TDaytypeID = array[1..255] of word;
19834: TDay, 'record Daynum : Word; MonthDate : word; DOWNum : word; M'
19835: +'onth : word; Week : word; DayCode : Word; end');
19836: CL.AddTypeS('TWeek', 'record WeekNum : word; WhichDays : TFIRSTLASTNUMBER; end');
19837: TMonth, 'record Month : word; Daycount : Word; WeekCount : Word'
19838: +'; WhichWeeks : TFIRSTLASTNUMBER; WhichDays : TFIRSTLASTNUMBER; end');
19839: TYear, 'record WeekCount : word; DayCount : Word; end');
19840: TKron, 'record ActiveYear : Word; IsInitialized : boolean; end');
19841: TYearExt, 'record Year : word; NumDays : word; NumWeeks : word;'
19842: +' LeapYear : boolean; YeartypeCount : word; end');
19843: TDateExt, 'record Year : word; DayofWeekNumber : word; DayName '
19844: +' string; MonthDay : Word; DayNumber : word; DaytypeCount : word; DaytypeI'
19845: +'D : TDaytypeID; MonthNumber : word; WeekNumber : word; Holiday : boolean; '
19846: +'ChurchDay : Boolean; Flagday : Boolean; end');
19847: TMonthExt, 'record Year : word; MonthNumber : word; MonthName :'
19848: +' string; FirstDay : word; LastDay : word; NumDays : word; NumWeeks : word;'
19849: +' FirstWeek : word; LastWeek : word; MonthImage : TMonthImage; end');
19850: TWeekExt, 'record Year : word; WeekNumber : word; FirstDay : word; LastDay : word; end');
19851: TForeignKey, 'record KeyName : string; KeyValue : Variant; end';
19852: TDaytypeDef, 'record AName : string; ADate : word; ARelDayTyp'
19853: +'e : word; AnOffset : integer; AFirstShowUp : word; ALastShowUp : word; ASh'
19854: +'owUpFrequency : word; AChurchDay : boolean; AHoliday : boolean; AFlagday :'
19855: +' boolean; AUserCalc : boolean; ATag : integer; end');
19856: SIRegister_TDaytype(CL);
19857: TWeekDay, '( Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday )';
19858: TWeekHolidays, 'set of TWeekDay');
19859: TOCEvent, '( ocYear, ocMonth, ocMonthnumber, ocWeek, ocWeeknumb'
19860: +'er, ocMonthDay, ocWeekday, ocDate, ocToday, ocCalcDaytype )');
19861: TCalcDaytypeEvent, 'Procedure ( Sender : TObject; Daytype : TDA'
19862: +'ytype; ADateExt : TDateExt; IsCurrentDate : boolean; var Accept : boolean)');
19863: TLoadDaytypeEvent, 'Procedure ( Sender : TObject; const Daytype'
19864: +'Def : TDaytypeDef; const DescKeys : String; ClassId : Integer; var LoadIt: boolean )';
19865: TSavedaytypeEvent, 'Procedure ( Sender : TObject; Daytype : TDA'
19866: +'ytype; var DescKeys : String; var ClassID : Integer; var SaveIt : boolean )';
19867: SIRegister_TKronos(CL);
19868: CL.AddClassN(CL.FindClass('TOBJECT'), 'EKronosError');
19869: //CL.AddDelphiFunction('Procedure Register');
19870: end;
19871:
19872: procedure SIRegister_MapFiles(CL: TPSPascalCompiler);
19873: begin
19874: CL.AddClassN(CL.FindClass('TOBJECT'), 'EMemoryMappedFile');
19875: CL.AddClassN(CL.FindClass('TOBJECT'), 'EMMEndOfFile');
19876: CL.AddTypeS('TFileOptions', '( foCreateNew, foCreateAlways, foOpenExisting, foOpenAlways,
19877: foTruncateExisting )');
19878: CL.AddTypeS('TmapFileType', '( ftUnspecified, ftRandomAccess, ftSequentialAccess )');
19879: SIRegister_TMapStream(CL);
19880: SIRegister_TTextMap(CL);
19881: SIRegister_TRecordMap(CL);
19882: end;
19883: procedure SIRegister_KGraphics(CL: TPSPascalCompiler);
19884: begin
19885: PNGHeader', 'String').SetString(' #137 ''PNG'' #13#10#26#10');
19886: MNGHeader', 'String').SetString(' #138 ''MNG'' #13#10#26#10');
19887: TKBrightMode', '( bsAbsolute, bsOfBottom, bsOfTop )';
19888: TKColorRec', 'record R : byte; G : byte; B : byte; A : Byte; Value : Cardinal; end');
19889: //PKColorRec', 'TKColorRec // will not work');
19890: TKDynColorRecs', 'array of TKColorRec');
19891: TKImageHeaderString', 'string');

```

```

19892: //TKPngImage', 'TPortableNetworkGraphic');
19893:   with CL.AddClassN(CL.FindClass('TLinarGraphic'), 'TPNGGraphic') do
19894:     TKPngImage', 'TPNGGraphic');
19895:     TKKString', 'WideString');
19896:   // TKString = WideString;
19897:   //TKPngImage', 'TPngImage');
19898: //TKPngImage', 'TPngObject');
19899:   TKTextAttribute', '( taCalcRect, taClip, taEndEllipsis, taFillRe'
19900:   +'ct, taFillText, taIncludePadding, taLineBreak, taPathEllipsis, taWordBreak'
19901:   +', taWrapText, taTrimWhiteSpaces, taStartEllipsis )');
19902:   TKTextAttributes', 'set of TKTextAttribute');
19903:   TKHAlign', '( halLeft, halCenter, halRight, halJustify )');
19904:   TKStretchMode', '( stmNone, stmZoomOutOnly, stmZoomInOnly, stmZoom )');
19905:   TKTextHAlign', 'TKHAlign');
19906:   TKVAlign', 'valTop, valCenter, valBottom )');
19907:   TKTextVAlign', 'TKVAlign');
19908:   TKButtonDrawState', '( bsUseThemes, bsDisabled, bsPressed, bsFocused, bsHot )');
19909:   TKButtonDrawStates', 'set of TKButtonDrawState');
19910:   SIRegister_TKGraphic(CL);
19911:   SIRegister_TKAlphaBitmap(CL);
19912:   SIRegister_TKMetafile(CL);
19913:   TKTextBoxFunction', '( tbfMeasure, tbfGetIndex, tbfGetRect, tbfDraw )');
19914:   SIRegister_TKTextBox(CL);
19915:   SIRegister_TKDragWindow(CL);
19916:   SIRegister_TKHintWindow(CL);
19917:   SIRegister_TKTextHint(CL);
19918:   SIRegister_TKGraphicHint(CL);
19919: //CL.AddDelphiFunction('Procedure BlendLine( Src, Dest : PKColorRecs; Count : Integer)');
19920: Function TKBrightColor( Color : TColor; Percent : Single; Mode : TKBrightMode) : TColor');
19921: Procedure CanvasGetScale( ACanvas : TCanvas; out MulX, MulY, DivX, DivY : Integer');
19922: Procedure CanvasResetScale( ACanvas : TCanvas)');
19923: Function CanvasScaled( ACanvas : TCanvas) : Boolean';
19924: Procedure CanvasSetScale( ACanvas : TCanvas; MulX, MulY, DivX, DivY : Integer)');
19925: Procedure CanvasSetOffset( ACanvas : TCanvas; OfsX, OfsY : Integer)');
19926: Function ColorRectToColor( Color : TKColorRec) : TColor');
19927: Function ColorToColorRec( Color : TColor) : TKColorRec');
19928: Function TKColorToGrayScale( Color : TColor) : TColor');
19929: Function CompareBrushes( ABrush1, ABrush2 : TBrush) : Boolean');
19930: Function CompareFonts( AFont1, AFont2 : TFont) : Boolean');
19931: Procedure CopyBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcX, SrcY : Integer)');
19932: Function CreateEmptyPoint : TPoint');
19933: Function CreateEmptyRect : TRect');
19934: Function CreateEmptyRgn : HRGN');
19935: Procedure DrawAlignedText( Canvas : TCanvas; var ARect : TRect; HAlign : TKHAlign; VAlign : TKVAlign;
  HPadding, VPadding : Integer; const AText : TKKString; BackColor : TColor; Attributes : TKTTextAttributes)');
19936: Procedure TKDrawButtonFrame( ACanvas:TCanvas;const ARect:TRect;AStates:TKButtonDrawStates)');
19937: Procedure DrawEdges(Canvas:TCanvas;const R:TRect;HighlightColor,ShadowColor:TColor;Flags: Cardinal);
19938: Procedure DrawFilledRectangle( Canvas : TCanvas; const ARect : TRect; BackColor : TColor)');
19939: Procedure DrawGradientRect( Canvas : TCanvas; const ARect : TRect; AStartColor, AEndColor : TColor;
  AColorStep : Integer; AHorizontal : Boolean)');
19940: Procedure ExcludeShapeFromBaseRect( var BaseRect : TRect; ShapeWidth, ShapeHeight : Integer; HAlign : TKHAlign;
  VAlign : TKVAlign; HPadding, VPadding : Integer; StretchMode : TKStretchMode; out Bounds,
  Interior : TRect)');
19941: Function ExtSelectClipRect(DC:HDC; ARect:TRect; Mode:Integer; var PrevRgn: HRGN): Boolean';
19942: Function ExtSelectClipRectEx(DC:HDC; ARect:TRect;Mode:Int;CurRgn,PrevRgn : HRGN): Boolean');
19943: Procedure FillAroundRect(ACanvas: TCanvas; const Boundary,Interior:TRect;BackColor:TColor);
19944: Function GetFontHeight( DC : HDC) : Integer');
19945: Function GetFontAscent( DC : HDC) : Integer');
19946: Function GetFontDescent( DC : HDC) : Integer');
19947: Function GDICheck( Value : Integer) : Integer');
19948: Function HorizontalShapePosition( AAlignment : TKHAlign; const ABoundary : TRect; const AShapeSize : TPoint) : Integer');
19949: Function ImageByType( const Header : TKImageHeaderString) : TGraphic');
19950: Function IntersectClipRectIndirect( DC : HDC; ARect : TRect) : Boolean');
19951: Function IsBrightColor( Color : TColor) : Boolean');
19952: Procedure LoadCustomCursor( Cursor : TCursor; const ResName : string)');
19953: Procedure TKLoadGraphicFromResource(Graphic: TGraphic;const ResName:string; ResType:PChar)');
19954: Function MakeColorRec( R, G, B, A : Byte) : TKColorRec');
19955: Function MakeColorRec5( Value : LongWord) : TKColorRec');
19956: Function PixelFormatFromBpp( Bpp : Cardinal) : TPixelFormat');
19957: Function TKRectInRegion( Rgn : HRGN; ARect : TRect) : Boolean');
19958: Function RgnCreateAndGet( DC : HDC) : HRGN');
19959: Procedure RgnSelectAndDelete( DC : HDC; Rgn : HRGN');
19960: Procedure RoundRectangle(ACanvas: TCanvas; const ARect: TRect;AXRadius,AYRadius: Integer)');
19961: Procedure SafeStretchDraw(ACanvas:TCanvas; ARect:TRect;AGraphic:TGraphic;ABackColor:TColor);
19962: Procedure SelectClipRect( DC : HDC; const ARect : TRect)');
19963: Procedure TKStretchBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcRect : TRect)');
19964: Function SwitchRGBToBGR( Value : TColor) : TColor');
19965: Procedure TranslateRectToDevice( DC : HDC; var ARect : TRect)');
19966: Function VerticalShapePosition( AAlignment : TKVAlign; const ABoundary : TRect; const AShapeSize : TPoint) : Integer');
19967: end;
19968:
19969: procedure SIRegister_umaxPipes(CL: TPPSPascalCompiler);
19970: begin
19971:   CL.AddConstantN('cShutdownMsg','String').SetString( 'shutdown pipe ');
19972:   CL.AddConstantN('cPipeFormat','String').SetString( '\\%s\pipe\%s');
19973:   SIRegister_TPipeServermax(CL);

```

```

19974:   SIRegister_TPipeClientmax(CL);
19975: end;
19976:
19977: procedure SIRegister_KControls(CL: TPSPascalCompiler);
19978: begin
19979:   CL.AddTypeS('TKColorArray', 'array of TColor');
19980:   TKPreviewColorIndex', 'Integer');
19981:   TKPrintOption', '( poCollected, poFitToPage, poMirrorMargins, poPa'
19982:     +'geNumbers, poPaintSelection, poTitle, poUseColor )');
19983:   TKPrintOptions', 'set of TKPrintOption');
19984:   TKPrintRange', '( kprAll, kprSelectedOnly, kprRange )');
19985:   TKPrintUnits', '( kpuMM, kpuCM, kpuInch, kpuHundredthInch )');
19986: //CL.AddConstantN('cBorderStyleDef','').SetString( bsSingle );
19987: 'cContentPaddingBottomDef','LongInt').SetInt( 0 );
19988: 'cContentPaddingLeftDef','LongInt').SetInt( 0 );
19989: CL.AddConstantN('cOptionsDef','LongInt').Value.ts32 := ord(poFitToPage) or ord(poPageNumbers) or
ord(poUseColor);
19990: //CL.AddConstantN('cRangeDef','').SetString( prAll );
19991: 'cScaleDef','LongInt').SetInt( 100 );
19992: 'cScaleMin','LongInt').SetInt( 10 );
19993: 'cScaleMax','LongInt').SetInt( 500 );
19994: //CL.AddConstantN('cUnitsDef','').SetString( puCM );
19995: //CL.AddConstantN('cPaperDef','').SetString( clWhite );
19996: //CL.AddConstantN('cBkGndDef','').SetString( clAppWorkSpace );
19997: //CL.AddConstantN('cBorderDef','').SetString( clBlack );
19998: //CL.AddConstantN('cSelectedBorderDef','').SetString( clNavy );
19999: CL.AddConstantN('ciPaper','LongInt').SetInt( TKPreviewColorIndex ( 0 ) );
20000: 'ciBkGnd','LongInt').SetInt( TKPreviewColorIndex ( 1 ) );
20001: 'ciBorder','LongInt').SetInt( TKPreviewColorIndex ( 2 ) );
20002: 'ciSelectedBorder','LongInt').SetInt( TKPreviewColorIndex ( 3 ) );
20003: // 'ciPreviewColorsMax','').SetString( ciSelectedBorder );
20004: 'cScrollNoAction','LongInt').SetInt( - 1 );
20005: 'cScrollDelta','LongInt').SetInt( - 2 );
20006: 'cPF_Dragging','LongWord').SetUInt( $00000001 );
20007: 'cFF_UpdateRange','LongWord').SetUInt( $00000002 );
20008: TKPreviewScaleMode', '( smScale, smPageWidth, smWholePage )');
20009: TKPreviewChangedEvent', 'Procedure ( Sender : TObject )';
20010: TKPrintMeasureInfo', 'record OutlineWidth : Integer; OutlineHeig'
20011:   ht : Integer; ControlHorzPageCount : Integer; ControlVertPageCount : Integ'
20012:   er; ExtraLeftHorzPageCount : Integer; ExtraLeftVertPageCount : Integer; Ex'
20013:   traRightHorzPageCount : Integer; ExtraRightVertPageCount : Integer; end');
20014: TKPrintStatus', '( kepsBegin, kepsNewPage, kepsEnd )');
20015: TKPrintNotifyEvent', 'Procedure (Sender:TObject; Status: TKPrintStatus; var Abort: Boolean )';
20016: TKPrintPaintEvent', 'Procedure ( Sender : TObject )';
20017: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKPrintPageSetup');
20018: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKPrintPreview');
20019: SIRegister_TKRect(CL);
20020: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKObjectList');
20021: SIRegister_TKObject(CL);
20022: //TKObjectClass', 'class of TKObject');
20023: SIRegister_TKObjectList(CL);
20024: SIRegister_TKCustomControl(CL);
20025: TKColorScheme', '( kcsNormal, kcsGrayed, kcsBright, kcsGrayScale )');
20026: TKColorIndex', 'Integer');
20027: TKColorData', 'record Index:TKColorIndex;Color:TColor; Default:TColor; Name:string; end');
20028: TKColorSpec', 'record Def : TColor; Name : string; end');
20029: SIRegister_TKCustomColors(CL);
20030: TKPrintMeasureEvent', 'Procedure ( Sender : TObject; var Info : TKPrintMeasureInfo )';
20031: SIRegister_TKPrintPageSetup(CL);
20032: SIRegister_TKPreviewColors(CL);
20033: SIRegister_TKPrintPreview(CL);
20034: Function InchesToValue( Units : TKPrintUnits; Value : Double ) : Double';
20035: Function ValueToInches( Units : TKPrintUnits; Value : Double ) : Double';
20036: end;
20037:
20038: procedure SIRegister_IdAntiFreezeBase(CL: TPSPascalCompiler);
20039: begin
20040:   CL.AddConstantN('ID_Default_TIdAntiFreezeBase_Active','Boolean').SetInt(1);
20041:   CL.AddConstantN('ID_Default_TIdAntiFreezeBase_ApplicationHasPriority','Boolean').SetInt(1);
20042:   CL.AddConstantN('ID_Default_TIdAntiFreezeBase_IdleTimeOut','LongInt').SetInt( 250 );
20043:   CL.AddConstantN('ID_Default_TIdAntiFreezeBase_OnlyWhenIdle','Boolean').SetInt( 250 );
20044:   SIRegister_TIdAntiFreezeBase(CL);
20045: end;
20046:
20047: procedure SIRegister_OverbyteIcsConApp(CL: TPSPascalCompiler);
20048: begin
20049:   CL.AddConstantN('WM_STARTUP','LongInt').SetInt( WM_USER + 789 );
20050:   SIRegister_TKeyboardThread(CL);
20051: //CL.AddTypeS('TConApplicationClass', 'class of TConApplication');
20052:   SIRegister_TConApplication(CL);
20053: end;
20054:
20055: procedure SIRegister_KMemo(CL: TPSPascalCompiler);
20056: begin
20057:   'cUndoLimitMin','LongInt').SetInt( 100 );
20058:   cUndoLimitMax','LongInt').SetInt( 10000 );
20059:   cUndoLimitDef','LongInt').SetInt( 1000 );
20060:   cScrollPaddingMin','LongInt').SetInt( 0 );
20061:   cScrollPaddingMax','LongInt').SetInt( 1000 );

```

```

20062: cScrollPaddingDef','LongInt').SetInt( 30 );
20063: cScrollSpeedMin','LongInt').SetInt( 50 );
20064: cScrollSpeedMax','LongInt').SetInt( 1000 );
20065: cScrollSpeedDef','LongInt').SetInt( 100 );
20066: //cBkGndDef,'').SetString( clWindow );
20067: //cInactiveCaretBkGndDef,'').SetString( clBlack );
20068: //cInactiveCaretSelBkGndDef,'').SetString( clBlack );
20069: //cInactiveCaretSelTextDef,'').SetString( clYellow );
20070: //cInactiveCaretTextDef,'').SetString( clYellow );
20071: //cSelBkGndDef,'').SetString( clGrayText );
20072: //cSelBkGndFocusedDef,'').SetString( clHighlight );
20073: //cSelTextDef,'').SetString( clHighlightText );
20074: //cSelTextFocusedDef,'').SetString( clHighlightText );
20075: ciBkGnd','LongInt').SetInt( TKColorIndex ( 0 ) );
20076: ciInactiveCaretBkGnd','LongInt').SetInt( TKColorIndex ( 1 ) );
20077: ciInactiveCaretSelBkGnd','LongInt').SetInt( TKColorIndex ( 2 ) );
20078: ciInactiveCaretSelText','LongInt').SetInt( TKColorIndex ( 3 ) );
20079: ciInactiveCaretText','LongInt').SetInt( TKColorIndex ( 4 ) );
20080: ciSelBkGnd','LongInt').SetInt( TKColorIndex ( 5 ) );
20081: ciSelBkGndFocused','LongInt').SetInt( TKColorIndex ( 6 ) );
20082: ciSelText','LongInt').SetInt( TKColorIndex ( 7 ) );
20083: ciSelTextFocused','LongInt').SetInt( TKColorIndex ( 8 ) );
20084: //ciMemoColorsMax','').SetString( ciSelTextFocused );
20085: cInvalidListID','LongInt').SetInt( - 1 );
20086: cHorzScrollStepDef','LongInt').SetInt( 4 );
20087: cVertScrollStepDef','LongInt').SetInt( 10 );
20088: //cHeight','LongInt').SetInt( 200 );
20089: //cWidth','LongInt').SetInt( 300 );
20090: cNewLineChar','Char').SetString( #$B6 );
20091: cSpaceChar','Char').SetString( #$B7 );
20092: cTabChar','Char').SetString( #$2192 );
20093: cBullet','Char').SetString( #$2022 );
20094: cSquareBullet','Char').SetString( #$25AB );
20095: cArrowBullet','Char').SetString( #$25BA );
20096: //cDefaultWordBreaks','String').SetString(' ' or '/ or '\ or ';' or ':' or '?' or '!');
20097: cRichText','String').SetString( 'Rich Text Format' );
20098: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKCustomMemo');
20099: CL.AddTypeS('TKMemoLinePosition', '( eolInside, eolEnd )');
20100: TKMemoBlockPosition', '( mbpText, mbpRelative, mbpAbsolute )');
20101: TKMemoState', '( elCaretCreated, elCaretVisible, elCareteUpdate, ,
20102: +elIgnoreNextChar, elModified, elMouseCapture, elOverwrite, elReadOnly )');
20103: TKMemoStates', 'set of TKMemoState');
20104: TKMemoUpdateReason', '( muContent, muExtent, muSelection, muSelectionScroll )');
20105: TKMemoUpdateReasons', 'set of TKMemoUpdateReason');
20106: SIRegister_TKMemoSparseItem(CL);
20107: SIRegister_TKMemoSparseList(CL);
20108: SIRegister_TKMemoSparseStack(CL);
20109: SIRegister_TKMemoDictionaryItem(CL);
20110: SIRegister_TKMemoDictionary(CL);
20111: TKMemoParaNumbering', '( pnuNone, pnuBullets, pnuArabic, pnuLett',
20112: +'erLo, pnuLetterHi, pnuRomanLo, pnuRomanHi )');
20113: SIRegister_TKMemoNumberingFormatItem(CL);
20114: SIRegister_TKMemoNumberingFormat(CL);
20115: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKMemoListLevels');
20116: SIRegister_TKMemoListLevel(CL);
20117: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKMemoList');
20118: SIRegister_TKMemoListLevels(CL);
20119: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKMemoListTable');
20120: SIRegister_TKMemoList(CL);
20121: TKMemoListChangedEvent', 'Procedure ( AList : TKMemoList; ALevel: TKMemoListLevel )');
20122: SIRegister_TKMemoListTable(CL);
20123: TKMemoScriptCapitals', '( tcaNone, tcaNormal, tcaSmall )');
20124: TKMemoScriptPosition', '( tpoNormal, tpoSuperscript, tpoSubscript )');
20125: SIRegister_TKMemoTextStyle(CL);
20126: TKMemoBlockWrapMode', '( wrAround, wrAroundLeft, wrAroundRight,
20127: +'wrTight, wrTightLeft, wrTightRight, wrTopBottom, wrNone, wrUnknown )');
20128: TKMemoBlockStyleChangedEvent', 'procedure ( Sender : TObject; AR'
20129: +'easons : TKMemoUpdateReasons )');
20130: SIRegister_TKMemoBlockStyle(CL);
20131: TKMemolineSpacingMode', '( lsmFactor, lsmValue )');
20132: SIRegister_TKMemoParaStyle(CL);
20133: SIRegister_TKMemoLine(CL);
20134: SIRegister_TKMemoLines(CL);
20135: SIRegister_TKMemoWord(CL);
20136: SIRegister_TKMemoWordList(CL);
20137: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKMemoBlocks');
20138: TKMemoMouseAction', '( maMove, maLeftDown, maLeftUp, maRightDown
20139: +' , maRightUp, maMidDown, maMidUp )');
20140: //TKMemoBlockClass', 'class of TKMemoBlock');
20141: SIRegister_TKMemoBlock(CL);
20142: SIRegister_TKMemoSingleton(CL);
20143: SIRegister_TKMemoTextBlock(CL);
20144: SIRegister_TKMemoHyperlink(CL);
20145: SIRegister_TKMemoParagraph(CL);
20146: SIRegister_TKMemoImageBlock(CL);
20147: SIRegister_TKMemoContainer(CL);
20148: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKMemoTable');
20149: CL.AddClassN(CL.FindClass('TOBJECT'), 'TKMemoTableRow');
20150: SIRegister_TKMemoTableCell(CL);

```

```

20151:  SIRегистrier_TKMemoTableRow(CL);
20152:  SIRегистrier_TKMemoTable(CL);
20153:  TKMemoUpdateEvent', 'Procedure ( Reasons : TKMemoUpdateReasons )';
20154:  SIRегистrier_TKMemoBlocks(CL);
20155:  SIRегистrier_TKMemoColors(CL);
20156:  TKMemoChangeKind', '( ckCaretPos, ckDelete, ckInsert )';
20157:  TKMemoUndoChangeEvent', 'Procedure ( Sender : TObject; ItemReason : TKMemoChangeKind )';
20158:  TKMemoChangeItem', 'record Blocks : TKMemoBlocks; Group : Cardinal;
20159:    al; GroupKind : TKMemoChangeKind; Inserted : Boolean; ItemKind : TKMemoCha'
20160:    ngeKind; Position : Integer; end';
20161:  //PKMemoChangeItem', '^TKMemoChangeItem // will not work';
20162:  SIRегистrier_TKMemoChangeList(CL);
20163:  TKMemoRTFString', 'string';
20164:  SIRегистrier_TKCustomMemo(CL);
20165:  SIRегистrier_TKMemo(CL);
20166:  SIRегистrier_TKMemoEditAction(CL);
20167:  SIRегистrier_TKMemoEditCopyAction(CL);
20168:  SIRегистrier_TKMemoEditCutAction(CL);
20169:  SIRегистrier_TKMemoEditPasteAction(CL);
20170:  SIRегистrier_TKMemoEditSelectAllAction(CL);
20171:  Function NewLineChar : TKkString';
20172:  Function SpaceChar : TKkString';
20173:  Function TabChar : TKkString';
20174: end;
20175:
20176: procedure SIRегистrier_OverbyteIcsTicks64(CL: TPSPascalCompiler);
20177: begin
20178:  CL.AddConstantN('ISODateMask','String').SetString( 'yyyy-mm-dd');
20179:  'ISODateTimeMask','String').SetString( 'yyyy-mm-ddT"hh:nn:ss');
20180:  'ISODateLongTimeMask','String').SetString( 'yyyy-mm-ddT"hh:nn:ss.zzz');
20181:  'ISOTimeMask','String').SetString( 'hh:nn:ss');
20182:  'LongTimeMask','String').SetString( 'hh:nn:ss:zzz');
20183:  'Ticks64PerDay','int64').SetInt64( 24 * 60 * 60 * 1000);
20184:  'Ticks64PerHour','int64').SetInt64( 60 * 60 * 1000);
20185:  'Ticks64PerMinute','int64').SetInt64( 60 * 1000);
20186:  'Ticks64PerSecond','int64').SetInt64( 1000);
20187:  CL.AddTypeS('TTicks64Mode', '( TicksNone, TicksAPI64, TicksPerf, TicksAPI32 )');
20188:  CL.AddDelphiFunction('Function IcsGetTickCount64 : int64');
20189:  Procedure IcsInitTick64( NewMode : TTicks64Mode );
20190:  Function IcsNowPC : TDateTime';
20191:  Procedure IcsAlignNowPC';
20192:  Function IcsLastBootDT : TDateTime';
20193:  Function IcsGetPerfCountsPerSec : int64';
20194:  Function IcsPerfCountCurrent : int64';
20195:  Function IcsPerfCountCurrMilli : int64';
20196:  Function IcsPerfCountToMilli( LI : int64 ) : int64';
20197:  Function IcsPerfCountGetMilli( startLI : int64 ) : int64';
20198:  Function IcsPerfCountGetMillStr( startLI : int64 ) : string';
20199:  Function IcsPerfCountToSecs( LI : int64 ) : integer';
20200:  Function IcsPerfCountGetSecs( startLI : int64 ) : integer';
20201:  Function IcsDiffTicks64( const StartTick, EndTick : int64 ) : int64';
20202:  Function IcsElapsedTicks64( const StartTick : int64 ) : int64';
20203:  Function IcsElapsedMsecs64( const StartTick : int64 ) : int64';
20204:  Function IcsElapsedSecs64( const StartTick : int64 ) : integer';
20205:  Function IcsElapsedMins64( const StartTick : int64 ) : integer';
20206:  Function IcsWaitingSecs64( const EndTick : int64 ) : integer';
20207:  Function IcsGetTrgMsecs64( const MilliSecs : int64 ) : int64';
20208:  Function IcsGetTrgSecs64( const DurSecs : integer ) : int64';
20209:  Function IcsGetTrgMins64( const DurMins : integer ) : int64';
20210:  Function IcsTestTrgTick64( const TrgTick : int64 ) : boolean';
20211:  Function IcsAddTrgMsecs64( const TickCount, MilliSecs : int64 ) : int64';
20212:  Function IcsAddTrgSecs64( const TickCount : int64; DurSecs : integer ) : int64';
20213: end;
20214:
20215: procedure SIRегистrier_OverbyteIcsSha1(CL: TPSPascalCompiler);
20216: begin
20217:  CL.AddConstantN('IcsSHA1Version','LongInt').SetInt( 800 );
20218:  CopyRight', 'String').SetString( ' IcsSHA1 (c) 2004-2012 F. Piette V8.00 ');
20219:  shaSuccess','LongInt').SetInt( 0 );
20220:  shaNull','LongInt').SetInt( 1 );
20221:  shaInputTooLong','LongInt').SetInt( 2 );
20222:  shaStateError','LongInt').SetInt( 3 );
20223:  SHA1HashSize','LongInt').SetInt( 20 );
20224:  CL.AddTypeS('uint32_t', 'LongWord');
20225:  CL.AddTypeS('uint8_t', 'Byte');
20226:  CL.AddTypeS('int_least16_t', 'LongInt');
20227:  CL.AddTypeS('SHA1DigestString', 'AnsiString');
20228:  CL.AddDelphiFunction('Function SHA1Reset( var context : SHA1Context ) : Integer');
20229:  Function SHA1Input( var context : SHA1Context; const message_array : String; message_array : PAnsiChar;
length : Cardinal ) : Integer';
20230:  Function SHA1Result( var context : SHA1Context; var Message_Digest : SHA1Digest ) : Integer');
20231:  Function SHA1ofStr( const s : AnsiString ) : SHA1DigestString';
20232:  Function SHA1ofBuf( const buf, buflen : Integer ) : SHA1DigestString';
20233:  Function SHA1ofStream( const strm : TStream ) : SHA1DigestString';
20234:  Function SHA1toHex( const digest : SHA1DigestString ) : String';
20235:  Function SHA1DigestToLowerHex( const Digest : SHA1Digest ) : String';
20236:  Function SHA1DigestToLowerHexA( const Digest : SHA1Digest ) : RawByteString';
20237:  Function SHA1DigestToLowerHexW( const Digest : SHA1Digest ) : UnicodeString';
20238:  Procedure HMAC_SHA1(const Data,DataLen:Integer;const Key,KeyLen:Int;out Digest:SHA1Digest);

```

```

20239: Function HMAC_SHA1_EX( const Data : AnsiString; const Key : AnsiString ) : AnsiString';
20240: end;
20241:
20242: procedure SIRegister_KEditCommon (CL: TPSPascalCompiler);
20243: begin
20244:   cCharMappingSize','LongInt').SetInt( 256 );
20245:   TKEditCommandk', '( ecNonek, ecLeftk, ecRightk, ecUpk, ecDownk, ecLinek',
20246:     Start, ecLineEndk, ecPageUpk, ecPageDownk, ecPageLeftk, ecPageRightk, ecPageTopk,
20247:     htk, ecSelUpk, ecSelDownk, ecSelLineStartk, ecSelLineEndk, ecSelPageUpk, ecSelPa',
20248:     gDownk, ecSelPageLeftk, ecSelPageRightk, ecSelPageTopk, ecSelPageBottomk, ecSe',
20249:     lEditorTopk, ecSelEditorBottomk, ecSelGotoXYk, ecScrollUpk, ecScrollDownk, ecSc',
20250:     rollLeftk, ecScrollRightk, ecScrollCenterk, ecUndok, ecRedok, ecCopyk, ecCutk, ec',
20251:     +'Pastek, ecInsertChark, ecInsertDigitsk, ecInsertStringk, ecInsertNewLinek, ecDe',
20252:     +'letLastChark, ecDeleteChark, ecDeleteBOLk, ecDeleteEOLk, ecDeleteLinek, ecSele',
20253:     +'ctAllk, ecClearAllk, ecClearIndexSelectionk, ecClearSelectionk, ecSearchk, ecRe',
20254:     +'placek, ecInsertModek, ecOverwriteModek, ecToggleModek, ecGotFocusk, ecLostFocu',
20255:     +'sk ')'); *)
20256: TKEditDisabledDrawStyle', '( eddBright, eddGrayed, eddNormal )');
20257: TKEditKey', 'record Key : Word; Shift : TShiftState; end');
20258: TKEditCommandAssignment', 'record Key : TKEditKey; Command : TKEditCommand; end');
20259: TKEditCommandMap', 'array of TKEditCommandAssignment');
20260: TKEditDropFilesEvent', 'Procedure ( Sender : TObject; X, Y : integer; Files : TStrings )';
20261: SIRegister_TKEditKeyMapping(CL);
20262: TKEditCharMapping', 'array of AnsiChar');
20263: //CL.AddTypeS('PKEditCharMapping', '^TKEditCharMapping // will not work');
20264: //CL.AddTypeS('TKEditOption', '( eoDropFiles, eoGroupUndo, eoUndoAfterSave, eos',
20265: //+'howFormatting, eoWantTab )');
20266: //CL.AddTypeS('TKEditOptions', 'set of TKEditOption');
20267: TKEditReplaceAction', '( eraCancel, eraYes, eraNo, eraAll )');
20268: TKEditReplaceTextEvent', 'Procedure ( Sender : TObject; const Te',
20269:   xtToFind, TextToReplace : string; var Action : TKEditReplaceAction )';
20270: TKEditSearchError', '( eseOk, eseNoDigitsFind, eseNoDigitsReplace, eseNoMatch )');
20271: TKEditSearchOption', '( esoAll, esoBackwards, esoEntireScope, es',
20272:   oFirstSearch, esoMatchCase, esoPrompt, esoSelectedOnly, esoTreatAsDigits, esoWereDigits )');
20273: TKEditSearchOptions', 'set of TKEditSearchOption');
20274: TKEditSearchData', 'record ErrorReason : TKEditSearchError; Opti',
20275:   ons : TKEditSearchOptions; SelStart : integer; SelEnd : Integer; TextToFin',
20276:   d : string; TextToReplace : string; end');
20277: // CL.AddTypeS('PKeditSearchData', '^TKEditSearchData // will not work');
20278: //cEditDisabledDrawStyleDef,'').SetString( eddBright),
20279: Function CreateDefaultKeyMapping : TKEditKeyMapping');
20280: Function DefaultCharMapping : TKEditCharMapping');
20281: Function DefaultSearchData : TKEditSearchData');
20282: end;
20283:
20284: unit UtilsMax4; unit uPSI_UtilsMax4;
20285:
20286: function AllDigitsDifferent(N: Int64): Boolean;
20287: procedure DecimalToFraction(Decimal: Extended; out FractionNumerator: Extended;
20288:                               out FractionDenominator: Extended; const AccuracyFactor: Extended);
20289: function ColorToHTML(const Color: TColor): string;
20290: function DOSCommand(const CommandLine: string; const CmdShow:integer;
20291:                       const WaitUntilComplete: Boolean; const WorkingDir: string = ''): Boolean;
20292: function GetDosOutput(CommandLine: string; Work: string = 'C:\'): string;
20293: procedure CaptureConsoleOutput(DosApp : string;AMemo : TMemo);
20294: function ExecuteCommandDOS(CommandLine:string):string;
20295: function DOSCommandRedirect(const CommandLine: string;
20296:                               const OutStream: Classes.TStream): Boolean; overload; //8
20297:
20298: procedure SendKeysToWindow(const HWnd: Windows.HWND; const Text: string);
20299: function IsRunningOnBattery: Boolean;
20300: function IsHexStr(const S: string): Boolean;
20301: function IsCharInSet(const Ch: Char; const Chars: TCharSet): Boolean;
20302: function StreamHasWatermark(const Stm: Classes.TStream;
20303:                               const Watermark: array of Byte): Boolean;
20304: function ReadBigEndianWord(Stm: Classes.TStream): Word;
20305: function DownloadURLToFile(const URL, FileName: string): Boolean;
20306: function ExtractURIQueryString(const URI: string): string;
20307: function GetBiosVendor: string;
20308: function GetIEVersionStr: string; //18
20309:
20310: function FloatToFixed(const Value: Extended; const DecimalPlaces: Byte;
20311:                        const SeparateThousands: Boolean): string;
20312: function IntToFixed(const Value: Integer;
20313:                       const SeparateThousands: Boolean): string;
20314: function Int64ToFixed(const Value: Int64;
20315:                        const SeparateThousands: Boolean): string;
20316: function IntToNumberText2(const Value: Integer): string; //22
20317:
20318: function IsLibraryInstalled2(const LibFileName: string): Boolean;
20319: function RemainingBatteryPercent: Integer;
20320: procedure SetLockKeyState(KeyCode: Integer; IsOn: Boolean);
20321: function IsLockKeyOn(const KeyCode: Integer): Boolean;
20322: procedure PostKeyEx322(const Key: Word; const Shift: Classes.TShiftState;
20323:                           const SpecialKey: Boolean = False);
20324: function TerminateProcessByID(ProcessID: Cardinal): Boolean;
20325: function GetWindowProcessName(const Wnd: Windows.HWND): string;
20326: function GetProcessName(const PID: Windows.DWORD): string;
20327: function GetWindowProcessID(const Wnd: Windows.HWND): Windows.DWORD;

```

```

20328: function IsAppResponding(const Wnd: Windows.HWND): Boolean;
20329: function IsTabletOS: Boolean;
20330: function ProgIDInstalled(const PID: string): Boolean;
20331: function GetProcessorName: string;
20332: function GetProcessorIdentifier: string; //36
20333: procedure EmptyKeyQueue;
20334: procedure TrimAppMemorySize;
20335: function GetEnvironmentBlockSize: Cardinal;
20336: function GetDefaultPrinterName: string; //40
20337: procedure ListDrives(const List: Classes.TStrings);
20338: procedure MultiSzToStrings(const MultiSz: PChar;
20339:                               const Strings: Classes.TStrings);
20340: function BrowseURL(const URL: string): Boolean;
20341: function IsValidURLProtocol(const URL: string): Boolean;
20342: function ExecAssociatedApp(const FileName: string): Boolean; //45
20343: function CheckInternetConnection(AHost: PAnsiChar): Boolean;
20344: function MakeSafeHTMLText(TheText: string): string;
20345: function RemoveURIQueryString(const URI: string): string;
20346: function GetRegistryString(const RootKey: Windows.HKEY;
20347:                               const SubKey, Name: string): string;
20348: procedure RefreshEnvironment2(const Timeout: Cardinal = 5000); //50
20349: function IsKeyPressed2(const VirtKeyCode: Integer): Boolean;
20350: function SizeOfFile64(const FileName: string): Int64;
20351: function IsHugeFile(const FileName: string): Boolean;
20352: function SetTransparencyLevel(const HWnd: Windows.HWND;
20353:                                 const Level: Byte): Boolean;
20354: function IsEqualResID(const R1, R2: PChar): Boolean; //55
20355: function GetGenericFileType(const FileNameOrExt: string): string;
20356: function GetFileType2(const FilePath: string): string;
20357: procedure ShowShellPropertiesDlg(const APath: string);
20358: function EllipsifyText(const AsPath: Boolean; const Text: string;
20359:                          const Canvas: Graphics.TCanvas; const MaxWidth: Integer): string;
20360: function CloneByteArray(const B: array of Byte): TBytes; //60
20361: procedure AppendByteArray(var B1: TBytes; const B2: array of Byte);
20362: functionIsUnicodeStream(const Stm: Classes.TStream): Boolean;
20363: function FileHasWatermark(const FileName: string;
20364:                             const Watermark: array of Byte; const Offset: Integer = 0): Boolean;
20365: function FileHasWatermarkAansi(const FileName: string;
20366:                                   const Watermark: AnsiString; const Offset: Integer = 0): Boolean;
20367: function IsASCIInputStream(const Stm: Classes.TStream; Count: Int64 = 0;
20368:                             BufSize: Integer = 8*1024): Boolean;
20369: function IsASCIIFile(const FileName: string; BytesToCheck: Int64 = 0;
20370:                        BufSize: Integer = 8*1024): Boolean; //66
20371: function BytesToAnsiString(const Bytes: SysUtils.TBytes; const CodePage: Word): String;
20372: function UnicodeStreamToWideString(const Stm: Classes.TStream): WideString;
20373: procedure WideStringToUnicodeStream(const Str: WideString;
20374:                                       const Stm: Classes.TStream);
20375: procedure GraphicToBitmap(const Src: Graphics.TGraphic;
20376:                            const Dest: Graphics.TBitmap; const TransparentColor: Graphics.TColor); //70
20377: 20378:
20379: function URIDecode(S: string; const IsQueryString: Boolean): string;
20380: function URIEncode(const S: string; const InQueryString: Boolean): string;
20381: function URLDecode(const S: string): string;
20382: function URLEncode(const S: string; const InQueryString: Boolean): string;
20383: function AllDigitsSame(N: Int64): Boolean;
20384: function GetDosOutput(CommandLine: string; Work: string = 'C:\'): string;
20385: procedure CaptureConsoleOutput(DosApp : string; AMemo : TMemo);
20386: function ExecuteCommandDOS(CommandLine:string):string;
20387: function DOSCommandRedirect(const CommandLine: string;
20388:                               const OutStream: Classes.TStream): Boolean; overload; //8
20389: procedure SendKeysToWindow(const HWnd: Windows.HWND; const Text: string);
20390: //75
20391: function FoldWrapText(const Line,BreakStr:string;BreakChars:TSysCharSet;MaxCol:Int):string;;
20392: function TextWrap(const Text: string; const Width, Margin: Integer): string;
20393: Function ShellExecuteX(Operation,FileName,Parameters,Directory: String; ShowCmd:Integer: Cardinal;');
20394: Function KeyboardStateToShiftState1(const KeyboardState: TKeyboardState): TShiftState;';
20395: Procedure SetCustomFormGlassFrame( const CustomForm : TCustomForm; const GlassFrame : TGlassFrame ');
20396: Function GetCustomFormGlassFrame( const CustomForm : TCustomForm ) : TGlassFrame';
20397: Procedure SetApplicationMainFormOnTaskBar(const Application:TApplication;Value: Boolean);
20398: Function GetApplicationMainFormOnTaskBar( const Application : TApplication ) : Boolean';
20399: function CompressWhiteSpace(const S: string): string;
20400: function IsASCIIDigit(const Ch: Char): Boolean;
20401: function CompareNumberStr(const S1, S2: string): Integer;
20402: procedure HexToBuf(HexStr: string; var Buf: string);
20403: function BufToHex(const Buf: string; const Size: Cardinal): string;
20404: Function DOSEexec( CommandLine : string; Work : string ) : string';
20405: procedure HexToStBuf(HexStr: string; var Buf: string;');
20406: function StrBufToHex(const Buf: string; const Size: Cardinal): string';
20407: function GetCharFromVirtualKey(AKey: Word): string;
20408: function GetParentProcessID(const PID: Windows.DWORD): Windows.DWORD;
20409: function FormInstanceCount2(AFormClass: Forms.TFormClass): Integer; //overload;
20410: function FormInstanceCount(const AFormClassName: string): Integer; //overload;
20411: function FindAssociatedApp(const Doc: string): string;
20412: function CreateShellLink2(const LinkFileName, AssocFileName, Desc, WorkDir, Args, IconFileName: string;
20413: const IconIdx: Integer): Boolean;
20414: function FileFromShellLink(const LinkFileName: string): string;
20415: function IsShellLink(const LinkFileName: string): Boolean;
20416: function ResourceIDToStr(const ResID: PChar): string;

```

```

20416: function IsEqualResID(const R1, R2: PChar): Boolean;
20417: function RecycleBinInfo(const Drive: Char; out BinSize, FileCount: Int64): Boolean;
20418: function FormInstanceCount2(AFormClass: TFormClass): Integer;');
20419: function FormInstanceCount(const AFormClassName: string): Integer;');
20420: function FindAssociatedApp(const Doc: string): string';
20421: function CreateShellLink2(const LinkFileName, AssocFileName, Desc, WorkDir, Args
20422: , IconFileName: string; const IconIdx: Integer): Boolean;
20423: function FileFromShellLink(const LinkFileName: string): string;
20424: function IsShellLink(const LinkFileName: string): Boolean;
20425: function ResourceIDToStr(const ResID: PChar): string;
20426: function IsEqualResID(const R1, R2: PChar): Boolean;
20427: function RecycleBinInfo(const Drive: Char; out BinSize, FileCount: Int64): Boolean;
20428: function SysImageListHandle(const Path:string; const WantLargeIcons:Boolean): THandle;
20429: function EmptyRecycleBin: Boolean;
20430: function ExploreFile(const Filename: string ): Boolean;
20431: function ExploreFolder(const Folder: string): Boolean;
20432: procedure ClearRecentDocs2;
20433: procedure AddToRecentDocs2(const FileName: string); //100
20434: function StringsToMultiSz(const Strings: Classes.TStrings; const MultiSz: PChar; const BufSize:
20435: Integer): Integer;
20436: procedure DrawTextOutline(const Canvas: TCanvas;const X,Y:Integer;const Text:string);
20437: function CloneGraphicAsBitmap(const Src: Graphics.TGraphic; const PixelFmt: Graphics.TPixelFormat;
20438: constTransparentColor: Graphics.TColor): Graphics.TBitmap;
20439: procedure InvertBitmap(const ABitmap: Graphics.TBitmap); //overload;
20440: procedure InvertBitmap2(const SrcBmp, DestBmp: Graphics.TBitmap); //overload;
20441: Function MakeFilenameInExePath( aFilename : Tfilename ) : Tfilename';
20442: Function YearOfDate2( DateTime : TDateTime) : Integer';
20443: Function MonthOfDate2( DateTime : TDateTime) : Integer';
20444: Function DayOfDate2( DateTime : TDateTime) : Integer';
20445: Function HourOfDateTime2( DateTime : TDateTime) : Integer';
20446: Function MinuteOfDateTime2( DateTime : TDateTime) : Integer';
20447: Function SecondOfDateTime2( DateTime : TDateTime) : Integer';
20448: Function IsLeapYear2( DateTime : TDateTime) : Boolean';
20449: Function DaysInMonth2( DateTime : TDateTime) : Integer';
20450: Function MakeUTCTime( DateTime : TDateTime) : TDateTime';
20451: Function MakeLocalTimeFromUTC( DateTime : TDateTime) : TDateTime';
20452: Function IsStandardTime : Boolean';
20453: Function UnixNow : Int64';
20454: Function NowString : string');
20455: Function MakeClosedTag( aTagName, aTagValue : string) : string';
20456: Function MakeOpenTag( aTagName, aTagAttributes : string) : string';
20457: Function MakeBold( Str : string) : string';
20458: Function MakeItalic( Str : string) : string';
20459: Function MakeUnderline( Str : string) : string';
20460: Function MakeStrikeout( Str : string) : string';
20461: Function MakeCenter( Str : string) : string';
20462: Function MakeParagraph( Str : string) : string';
20463: Function MakeCode( Str : string) : string';
20464: Function MakeOption( aValue, aText : string) : string';
20465: Function MakeHTMLFontSize( Str : string; SizeParam : string) : string';
20466: Function AddQuotes2( Str : string) : string';
20467: Function AddSingleQuotes( Str : string) : string';
20468: Function MakeLink( URL, name : string) : string';
20469: Function MakeLinkTarget( URL, name, Target : string) : string';
20470: Function MakeMailTo( Address, name : string) : string';
20471: Function HTMLToDelphiColor( S : string) : TColor';
20472: Function ColorToHTMLHex( Color : TColor) : string';
20473: Function GetStringFromRes( ResName : string) : string';
20474: Function EscapeText( sText : string) : string';
20475: Function EncodeForXML( const aString : string) : string';
20476: Function IsStringAlpha( Str : string) : Boolean';
20477: Function IsStringNumber( Str : string) : Boolean';
20478: Function EnsurePrefix( aPrefix, aText : string) : string';
20479: Function StringToAcceptableChars( S : string; AcceptableChars : TCharSet) : string';
20480: Function StringIsAcceptable( S : string; AcceptableChars : TCharSet) : Boolean';
20481: Function ValidateEmailAddress( aEmail : string) : Boolean';
20482: Function FirstChar( Str : string) : Char';
20483: Function LastChar( Str : string) : Char';
20484: Function StringIsEmpty( Str : string) : Boolean';
20485: Function StringIsNotEmpty( Str : string) : Boolean';
20486: Function StringHasSpacesInMiddle( Str : string) : Boolean';
20487: Function StringContains( aString : string; aSubStr : string) : Boolean';
20488: Function SpacesToUnderscore( S : string) : string';
20489: Function SpacesToPluses( Str : string) : string';
20490: Function SwapString( Str : string) : string';
20491: Function MakeCopyrightNotice( aCopyrightHolder : string) : string';
20492: Procedure WriteStringToFile( aStr : string; aFilename : Tfilename)';
20493: //Function WinExecAndWait32( Path : PChar; Visibility : Word) : Integer';
20494: Function WinExecAndWait32V2( Filename : string; Visibility : Integer) : DWORD';
20495: Function WindowsExit( RebootParam : Longword) : Boolean';
20496: Function GetVersionInfo2 : string';
20497: Function VersionString( aPrefix : string; aUseColon : Boolean) : string';
20498: Function CreateTempFileName( aPrefix : string) : string';
20499: Function GetWindowsTempDir : string';
20500: Function GetWindowsDir2 : string';
20501: Function GetSystemDir2 : string';
20502: Function GetSpecialFolderLocation( aFolderType : Integer) : string';
20503: Function GetTextFileContents( aFilename : Tfilename) : string';

```

```

20504: Function GetBDSDir( aVersion : Integer ) : string';
20505: Function CaptionMessageDlg( const aCaption : string; const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx : Longint ) : Integer';
20506: Function StreamFileCopy(const SourceFilename,TargetFilename:string; KeepDate:Boolean):Integer;
20507: Function MakePercentString( f : Double ) : string';
20508: Procedure DumpKey( var aKey : Char )';
20509: Function ValidateKey2(var aKey:Char; AcceptableKeys:TCharSet;KillTheKey: Boolean) : Boolean';
20510: Function MakeInterbaseString( aHostName, aFilename : string ) : string';
20511: Function ParseToken( const S : string; var FromPos : Integer; Delimiter : Char ) : string';
20512: Procedure MatchBounds( MovedControl, TemplateControl : TControl )';
20513: Procedure LockWidth( aControl : TControl )';
20514: Procedure LockHeight( aControl : TControl )';
20515: Procedure LockBounds( aControl : TControl )';
20516: Function TruncateFilename( aCanvas : TCanvas; aRect : TRect; aFilename : string; aMargin : Integer ) : string';
20517: Function PointInRect2( const R : TRect; const P : TPoint ) : Boolean;';
20518: Function PointInRect3( const R : TRect; const X, Y : Integer ) : Boolean;';
20519: Procedure VariantToStream2( const V : OLEVariant; Stream : TStream )';
20520: Procedure StreamToVariant2( Stream : TStream; var V : OLEVariant )';
20521: Procedure AssignDocument( Browser : TWebBrowser; Text : string )';
20522: Procedure LoadStreamToWebBrowser( WebBrowser : TWebBrowser; Stream : TStream )';
20523: Procedure SaveWebBrowserSourceToStream( Document : IDispatch; Stream : TStream )';
20524: Procedure GetStylesFromBrowser( aBrowser : TWebBrowser; aStyles : TStrings )';
20525: CL.AddTypeS('TnxBits', 'Integer');
20526: CL.AddTypeS('TDayHours', 'Integer');
20527: CL.AddClassN(CL.FindClass('TOBJECT'), 'NixUtilsException');
20528: Function REG_CURRENT_VERSION : string';
20529: Function RegisteredOwner : string';
20530: Function RegisteredCompany : string';
20531: Function GetLocalComputerName2 : string';
20532: Function GetLocalUserName2 : string';
20533: Function DeleteToRecycleBin(WindowHandle: HWND; Filename:string; Confirm:Boolean): Boolean';
20534: Function RemoveBackSlash2( const Dir : string ) : string';
20535: Function EnsureBackSlash( aPath : string ) : string';
20536: Function EnsureForwardSlash( aPath : string ) : string';
20537: Function RemoveLeadingSlash( aPath : string ) : string';
20538: Function SameDirectories( aDir1, aDir2 : TFilename ) : Boolean';
20539: Function EnsureExtensionHasLeadingPeriod( aExtension : string ) : string';
20540: Function RemoveExtension( afilename : string ) : string';
20541: Function GetModuleFileNameStr : string';
20542: Function ModulePath : string';
20543: Function IniFileName : string';
20544: Function MakeFilenameInExePath( afilename : TFilename ) : TFilename';
20545: Function IsBitSet( Bits : Integer; BitToSet : TnxBits ) : Boolean';
20546: Function SetBit6( Bits : Integer; BitToSet : TnxBits ) : Integer';
20547: Function UnSetBit( Bits : Integer; BitToSet : TnxBits ) : Integer';
20548: Function FlipBit( Bits : Integer; BitToSet : TnxBits ) : Integer';
20549: Function / FontToOleFont
20550: Function / OleFontToFont
20551: function GetCachedFileFromURL(strUL: string; var strLocalFile: string): boolean;
20552: function IAddrToHostName(const IP: string): string;
20553: function GetIEHandle(WebBrowser: TWebbrowser; ClassName: string): HWND;
20554: function GetTextFromHandle(WinHandle: THandle): string;
20555: procedure Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
20556: function FillWebForm(WebBrowser: TWebBrowser; FieldName: string; Value: string): Boolean;
20557: procedure WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode: string);
20558: function NetSend(dest, Source, Msg: string): Longint; overload;
20559:
20560: end;
20561:
20562: procedure SIRegister_UWANTUtils(CL: TPPascalCompiler);
20563: begin
20564:   TErrorSeverity', '(errwantNone, errwantInfo, errwantHint, errwantWarning, errwantError, errwantFatal ))';
20565:   Function ErrTag( Sev : TErrorSeverity ) : String';
20566:   Function MakeInt64( LowInt32, HiInt32 : Cardinal ) : Int64';
20567:   Function FolderContent( const APath : String ) : INode';
20568:   Function FindFiles2( const AFolder, AMask : String; Attrs : Integer; AList : TStrings; ASearchSubdirs : Boolean ) : Boolean';
20569:   Function ErrorSeverityStr( ASeverity : TErrorSeverity ) : String';
20570:   Function IsParameterOptional( AParamDef : INode ) : Boolean';
20571:   Function ExtractMethodName( const S : String ) : String';
20572:   Function LineNoOf( ANode : INode ) : Integer';
20573:   Function ColNoOf( ANode : INode ) : Integer';
20574:   Function WANTBoolToStr( AValue : Boolean ) : String';
20575:   Function WANTStrToBool( const AValue : String; dEFAULT : bBOOLEAN ) : Boolean';
20576:   Function StrDefault( const S, Default : String ) : String';
20577:   Function PluralNoun( Count : Integer; const S : String ) : String';
20578:   Function IsUnqualifiedName( const AName : String ) : Boolean';
20579:   Function IsModuleTag( const ATag : String ) : Boolean';
20580:   Function IsCallableTag( const ATag : String ) : Boolean';
20581:   Function IsConditionalTag( const ATag : String ) : Boolean';
20582:   Function IsMethodTag( const ATag : String ) : Boolean';
20583:   Function StripDriveFromPath( const AFileName : String ) : String';
20584:   Function TotalTime( NumTicks : Int64 ) : String';
20585:
20586:
20587: procedure SIRegister_IdNNTPServer(CL: TPPascalCompiler);
20588: begin
20589:   TIdNNTPAuthType', '( atUserPass, atSimple, atGeneric )';

```

```

20590: TIdNNTPAuthTypes', 'set of TIdNNTPAuthType');
20591: SIRegister_TIdNNTPThread(CL);
20592: CL.AddTypeS('TIdNNTPOnAuth', 'Procedure ( AThread : TIdNNTPThread; var VAccept : Boolean)');
20593: TIdNNTPOnNewGroupsList', 'Procedure ( AThread : TIdNNTPThread; c'
20594: +'onst ADateStamp : TDateTime; const ADistributions : String)');
20595: TIdNNTPOnNewNews', 'Procedure ( AThread : TIdNNTPThread; const N'
20596: +'ewsgroups : String; const ADateStamp : TDateTime; const ADistributions : String)');
20597: TIdNNTPOnIHaveCheck', 'Procedure ( AThread : TIdNNTPThread; cons'
20598: +'t AMsgID : String; VAccept : Boolean)');
20599: TIdNNTPOnArticleByNo', 'Procedure ( AThread : TIdNNTPThread; const AMsgNo : Integer)');
20600: TIdNNTPOnArticleByID', 'Procedure ( AThread : TIdNNTPThread; const AMsgID : string)');
20601: TIdNNTPOnCheckMsgNo', 'Procedure ( AThread : TIdNNTPThread; cons'
20602: +'t AMsgNo : Integer; var VMsgID : string)');
20603: TIdNNTPOnCheckMsgID', 'Procedure ( AThread : TIdNNTPThread; cons'
20604: +'t AMsgId : string; var VMsgNo : Integer)');
20605: TIdNNTPOnMovePointer', 'Procedure ( AThread : TIdNNTPThread; var'
20606: +' AMsgNo : Integer; var VMsgID : string)');
20607: TIdNNTPOnPost', 'Procedure ( AThread : TIdNNTPThread; var VPostO'
20608: +'k : Boolean; var VErrorText : string)');
20609: TIdNNTPOnSelectGroup', 'Procedure ( AThread : TIdNNTPThread; con'
20610: +'st AGroup : string; var VMsgCount : Integer; var VMsgFirst : Integer; var '
20611: +'VMsgLast : Integer; var VGroupExists : Boolean)');
20612: TIdNNTPOnCheckListGroup', 'Procedure ( AThread : TIdNNTPThread; '
20613: +'const AGroup : string; var VCanJoin : Boolean; var VFirstArticle : Integer)');
20614: TIdNNTPOnXHdr', 'Procedure ( AThread : TIdNNTPThread; const AHea'
20615: +derName: String; const AMsgFirst: Integer; const AMsgLast:Integer;const AMsgID:String)');
20616: TIdNNTPOnXOver', 'Procedure ( AThread : TIdNNTPThread; const AMs'
20617: +'gFirst : Integer; const AMsgLast : Integer)');
20618: TIdNNTPOnXPat', 'Procedure ( AThread : TIdNNTPThread; const AHea'
20619: +'derName : String; const AMsgFirst : Integer; const AMsgLast : Integer; con'
20620: +'st AMsgID : String; const AHeaderPattern : String)');
20621: TIdNNTPOnAuthRequired', 'Procedure ( AThread : TIdNNTPThread; co'
20622: +'nst ACommand, AParams : string; var VRequired : Boolean)');
20623: TIdNNTPOnListPattern', 'Procedure ( AThread : TIdNNTPThread; con'
20624: +'st AGroupPattern : String)');
20625: SIRegister_TIdNNTPServer(CL);
20626: end;
20627:
20628: procedure SIRegister_OverbyteIcsAsn1Utils(CL: TPPSPascalCompiler);
20629: begin
20630:   'ASN1UtilsVersion','LongInt').SetInt( 101);
20631:   CL.AddConstantN('ASN1_BOOL','LongWord').SetUInt( $01);
20632:   'ASN1_INT','LongWord').SetUInt( $02);
20633:   'ASN1_OCTSTR','LongWord').SetUInt( $04);
20634:   'ASN1_NULL','LongWord').SetUInt( $05);
20635:   'ASN1_OBJID','LongWord').SetUInt( $06);
20636:   'ASN1_ENUM','LongWord').SetUInt( $0a);
20637:   'ASN1_SEQ','LongWord').SetUInt( $30);
20638:   'ASN1_SETOF','LongWord').SetUInt( $31);
20639:   'ASN1_IPADDR','LongWord').SetUInt( $40);
20640:   'ASN1_COUNTER','LongWord').SetUInt( $41);
20641:   'ASN1_GAUGE','LongWord').SetUInt( $42);
20642:   'ASN1_TIMETICKS','LongWord').SetUInt( $43);
20643:   'ASN1_OPAQUE','LongWord').SetUInt( $44);
20644:   Function ASNEncOIDItem2( Value : Integer) : AnsiString';
20645:   Function ASNDecOIDItem2( var Start : Integer; const Buffer : AnsiString) : Integer';
20646:   Function ASNEncLen2( Len : Integer) : AnsiString';
20647:   Function ASNDeclen2( var Start : Integer; const Buffer : AnsiString) : Integer';
20648:   Function ASNEncInt2( Value : Integer) : AnsiString';
20649:   Function ASNEncUint2( Value : Integer) : AnsiString';
20650:   Function ASNOBJECT2( const Data : AnsiString; ASNTYPE : Integer) : AnsiString';
20651:   Function ASNItem2( var Start:Integer; const Buffer:AnsiString; var ValueType:Integer): AnsiString';
20652:   Function MibToId2( Mib : String) : AnsiString';
20653: //Function MibToId1( Mib : AnsiString) : AnsiString;';
20654:   Function IdToMib2( const Id : AnsiString) : String';
20655:   Function IntMibToStr2( const Value : AnsiString) : AnsiString';
20656:   Function ASNdump2( const Value : AnsiString) : String';
20657: end;
20658:
20659:   const SM_MEDIACENTER = 87; // metrics flag not defined in Windows unit
20660:   SM_TABLETPC = 86;
20661:
20662:   http://snippets.delphidabbler.com/#
20663:
20664: procedure SIRegister_wmiserv(CL: TPPSPascalCompiler);
20665: begin
20666:   CL.AddConstantN('EOAC_NONE','LongInt').SetInt( 0);
20667:   'RPC_C_AUTHN_WINNT','LongInt').SetInt( 10);
20668:   'RPC_C_AUTHZ_NONE','LongInt').SetInt( 0);
20669:   'RPC_E_CHANGED_MODE','LongInt').SetInt( - 2147417850);
20670:   Function WMISStart : ISWBemLocator)';
20671:   Function WMIMConnect(WBemLocator:ISWBemLocator; Server,account,password:string):ISWBemServices';
20672:   Function WMIEexecQuery( WBemServices : ISWBemServices; query : string ) : ISWbemObjectSet';
20673:   Function WMIRowFindFirst( ObjectSet: ISWBemObjectSet; var ENum: IEnumVariant; var tempobj : OleVariant) : boolean';
20674:   Function WMIRowFindNext( ENum : IEnumVariant; var tempobj : OleVariant) : boolean';
20675:   Function WMIColFindFirst( var propEnum: IEnumVariant; var tempObj : OleVariant) : boolean';
20676:   Function WMIColFindNext( propEnum : IEnumVariant; var tempobj : OleVariant) : boolean';
20677:   Function WMIGetValue( wbemservices: ISWBemServices; tablename, fieldname : string ) : string';

```

```

20678: Function WMICConvValue( tempobj : OleVariant; var keyname : string ) : string');
20679: end;
20680:
20681: procedure SIRRegister_RegSvrUtils(CL: TPSPascalCompiler);
20682: begin
20683:   SIRRegister_ERegistryException(CL);
20684:   Function RegOpenKey2( Key : HKey; const SubKey : string ) : HKey');
20685:   Function RegGetKey( Key : HKey; const SubKey : string ) : string');
20686:   Function RegCanOpenKey( Key : HKey; const SubKey : string; var OutKey : HKey ) : Boolean');
20687:   Function RegKeyExists2( Key : HKey; const SubKey : string ) : Boolean');
20688:   Function RegCloseAndNilKey( var Key : HKey ) : Boolean');
20689:   Function RegQuerySubKeyCount( Key : HKey ) : Integer');
20690:   Function RegEnumKey2( Key : HKey; Index : Integer; var Value : string ) : Boolean');
20691:   Function RegQueryKey( Key : HKey; const SubKey : string; var Value : string ) : Boolean');
20692:   Function RegGetValueEx( Key : HKey; const ValName : string ) : string');
20693:   Procedure ErrorFmt( const Ident : string; const Args : array of const );
20694:   Procedure FmtError( const Ident : string; const Args : array of const );
20695: end;
20696: end;
20697:
20698: procedure SIRRegister_osFileUtil(CL: TPSPascalCompiler); //Linux
20699: begin
20700:   Function OsDOS2UnixFileAttributes( Attr : LongWord ) : LongWord');
20701:   Function OsUnix2DosFileAttributes( Attr : LongWord ) : LongWord');
20702:   Function OsUnixFileTimeToDate( UnixTime : LongInt ) : TDateTime');
20703:   Function OsDateTimeToUnixFileTime( DateTime : TDateTime ) : LongInt');
20704:   Function OsDosFileTimeToDate( DosTime : LongInt ) : TDateTime');
20705:   Function OsDateTimeToDosFileTime( Value : TDateTime ) : LongInt');
20706:   Function OsFileTimeToLocalFileTime( FileTime : LongInt ) : LongInt');
20707:   Function OsLocalFileTimeToFileTime( FileTime : LongInt ) : LongInt');
20708: end;
20709:
20710: procedure SIRRegister_TWebBrowser(CL: TPSPascalCompiler);
20711: begin
20712:   //with RegClassS(CL,'ToleControl', 'TWebBrowser') do
20713:   with CL.AddClassN(CL.FindClass('ToleControl'), 'TWebBrowser') do begin
20714:     RegisterMethod('Procedure GoBack');
20715:     Procedure GoForward';
20716:     Procedure GoHome';
20717:     Procedure GoSearch';
20718:     Procedure Navigate7( const URL : WideString );'
20719:     Procedure Navigate8( const URL : WideString; const Flags : OleVariant );'
20720:     Procedure Navigate9( const URL:WideString;const Flags:OleVariant;const TargetFrameName: OleVariant );'
20721:     Procedure Navigate10( const URL : WideString; const Flags : OleVariant; const TargetFrameName : OleVariant; var PostData : OleVariant );'
20722:     Procedure Navigate11( const URL : WideString; const Flags : OleVariant; const TargetFrameName : OleVariant; var PostData : OleVariant; const Headers : OleVariant );'
20723:     Procedure Refresh';
20724:     Procedure Refresh212; '
20725:     Procedure Refresh213( var Level : OleVariant );'
20726:     Procedure Stop';
20727:     Procedure Quit';
20728:     Procedure ClientToWindow( var pcx : SYSINT; var pcy : SYSINT );
20729:     Procedure PutProperty( const Property_ : WideString; vtValue : OleVariant );
20730:     Function GetProperty( const Property_ : WideString ) : OleVariant );
20731:     Procedure Navigate214( var URL : OleVariant );'
20732:     Procedure Navigate215( var URL : OleVariant; var Flags : OleVariant );'
20733:     Procedure Navigate216( var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVar );
20734:     Procedure Navigate217( var URL : OleVariant; var Flags : OleVariant; var TargetFrameName : OleVariant; var PostData : OleVariant );'
20735:     Procedure Navigate218( var URL : OleVariant; var Flags : OleVariant; var TargetFrameName : OleVariant; var PostData : OleVariant; var Headers : OleVariant );'
20736:     Function QueryStatusWB( cmdID : OLECMDID ) : OLECMDF';
20737:     Procedure ExecWB19( cmdID : OLECMDID; cmdexecopt : OLECMDEXECOPT );'
20738:     Procedure ExecWB20( cmdID : OLECMDID; cmdexecopt : OLECMDEXECOPT; var pvaIn : OleVariant );'
20739:     Procedure ExecWB21( cmdID : OLECMDID; cmdexecopt : OLECMDEXECOPT; var pvaIn : OleVariant; var pvaOut : OleVariant );'
20740:     Procedure ShowBrowserBar22( var pvaClSID : OleVariant );'
20741:     Procedure ShowBrowserBar23( var pvaClSID : OleVariant; var pvarShow : OleVariant );'
20742:     Procedure ShowBrowserBar24( var pvaClSID : OleVariant; var pvarShow : OleVariant; var pvarSize : OleVariant );'
20743:     ControlInterface', 'IWebBrowser2', iptr);
20744:     RegisterProperty('DefaultInterface', 'IWebBrowser2', iptr);
20745:     Application', 'IDispatch', iptr);
20746:     Parent', 'IDispatch', iptr);
20747:     Container', 'IDispatch', iptr);
20748:     Document', 'IDispatch', iptr);
20749:     TopLevelContainer', 'WordBool', iptr);
20750:     type_, 'WideString', iptr);
20751:     LocationName', 'WideString', iptr);
20752:     LocationURL', 'WideString', iptr);
20753:     Busy', 'WordBool', iptr);
20754:     Name', 'WideString', iptr);
20755:     FullName', 'WideString', iptr);
20756:     Path', 'WideString', iptr);
20757:     Visible', 'WordBool', iptrw);
20758:     StatusBar', 'WordBool', iptrw);
20759:     StatusText', 'WideString', iptrw);
20760:    ToolBar', 'Integer', iptrw);

```

```

20761:   MenuBar', 'WordBool', iptrw);
20762:   FullScreen', 'WordBool', iptrw);
20763:   Offline', 'WordBool', iptrw);
20764:   Silent', 'WordBool', iptrw);
20765:   RegisterAsBrowser', 'WordBool', iptrw);
20766:   RegisterAsDropTarget', 'WordBool', iptrw);
20767:   TheaterMode', 'WordBool', iptrw);
20768:   AddressBar', 'WordBool', iptrw);
20769:   Resizable', 'WordBool', iptrw);
20770:   OnStatusTextChange', 'TWebBrowserStatusTextChange', iptrw);
20771:   OnProgressChange', 'TWebBrowserProgressChange', iptrw);
20772:   OnCommandStateChange', 'TWebBrowserCommandStateChange', iptrw);
20773:   OnDownloadBegin', 'TNotifyEvent', iptrw);
20774:   OnDownloadComplete', 'TNotifyEvent', iptrw);
20775:   OnTitleChange', 'TWebBrowserTitleChange', iptrw);
20776:   OnPropertyChange', 'TWebBrowserPropertyChange', iptrw);
20777:   OnBeforeNavigate2', 'TWebBrowserBeforeNavigate2', iptrw);
20778:   OnNewWindow2', 'TWebBrowserNewWindow2', iptrw);
20779:   OnNavigateComplete2', 'TWebBrowserNavigateComplete2', iptrw);
20780:   OnDocumentComplete', 'TWebBrowserDocumentComplete', iptrw);
20781:   OnQuit', 'TNotifyEvent', iptrw);
20782:   OnVisible', 'TWebBrowserOnVisible', iptrw);
20783:   OnToolBar', 'TWebBrowserOnToolBar', iptrw);
20784:   OnMenuBar', 'TWebBrowserOnMenuBar', iptrw);
20785:   OnStatusBar', 'TWebBrowserOnStatusBar', iptrw);
20786:   OnFullScreen', 'TWebBrowserOnFullScreen', iptrw);
20787:   OnTheaterMode', 'TWebBrowserOnTheaterMode', iptrw);
20788:   OnWindowSetResizable', 'TWebBrowserWindowSetResizable', iptrw);
20789:   OnWindowSetLeft', 'TWebBrowserWindowSetLeft', iptrw);
20790:   OnWindowSetTop', 'TWebBrowserWindowSetTop', iptrw);
20791:   OnWindowSetWidth', 'TWebBrowserWindowSetWidth', iptrw);
20792:   OnWindowSetHeight', 'TWebBrowserWindowSetHeight', iptrw);
20793:   OnWindowClosing', 'TWebBrowserWindowClosing', iptrw);
20794:   OnClientToHostWindow', 'TWebBrowserClientToHostWindow', iptrw);
20795:   OnSetSecureLockIcon', 'TWebBrowserSetSecureLockIcon', iptrw);
20796:   OnFileDownload', 'TWebBrowserFileDownload', iptrw);
20797:   OnNavigateError', 'TWebBrowserNavigateError', iptrw);
20798:   OnPrintTemplateInstantiation', 'TWebBrowserPrintTemplateInstantiation', iptrw);
20799:   OnPrintTemplateTeardown', 'TWebBrowserPrintTemplateTeardown', iptrw);
20800:   OnUpdatePageStatus', 'TWebBrowserUpdatePageStatus', iptrw);
20801:   OnPrivacyImpactedStateChange', 'TWebBrowserPrivacyImpactedStateChange', iptrw);
20802: end;
20803: end;
20804:
20805: procedure SIRegister_xutils(CL: TPSFCompiler);
20806: begin
20807:   //CL.AddTypeS('PshortString', '^ShortString // will not work');
20808:   'EXIT_DOSERROR','LongInt').SetInt( 2 );
20809:   'EXIT_ERROR','LongInt').SetInt( 1 );
20810:   'adCmdTxt','LongInt').SetInt($00000001);
20811:   'adExecNoRecords','LongInt').SetInt($00000080);
20812:   {** Byte order mark: UTF-8 encoding signature }
20813:   ('BOM_UTF8','String').SetString( #$EF#$BB#$BE);
20814:   {** Byte order mark: UTF-8 encoding signature }
20815:   BOM_UTF8      = #$EF#$BB#$BE;
20816:   {** Byte order mark: UCS-4 big endian encoding signature }
20817:   BOM_UTF32_BE = #00#00#$FE#$FF;
20818:   {** Byte order mark: UCS-4 little endian encoding signature }
20819:   BOM_UTF32_LE = #$FF#$FE#00#00;
20820:   BOM_UTF16_BE = #$FE#$FF;
20821:   BOM_UTF16_LE = #$FF#$FE;
20822:   //const adCmdTxt = $00000001;
20823:   //adExecNoRecords = $00000080;
20824:   Function AnsiFileExists( const FName : string ) : Boolean';
20825:   Function AnsiDirectoryExists( DName : string ) : Boolean';
20826:   Procedure SwapLong2( var x : longword );
20827:   Procedure SwapWord2( var x : word );
20828:   Function UpString( s : string ) : string';
20829:   Function LowString( s : string ) : string';
20830:   Function AddDoubleQuotes( s : string ) : string';
20831:   Function RemoveDoubleQuotes( s : string ) : string';
20832:   Procedure StreamErrorProcedure( var S : TStream );
20833:   Function StrToken( var Text : String; Delimiter : Char; UseQuotes : boolean ) : String';
20834:   Function StrGetNextLine( var Text : String ) : String';
20835:   Function TrimLeftx( const S : string ) : string';
20836:   Function TrimRightx( const S : string ) : string';
20837:   Function hexstr( val : longint; cnt : byte ) : string';
20838:   Function decstr( val : longint; cnt : byte ) : string';
20839:   Function decstrnsigned( l : longword; cnt : byte ) : string';
20840:   Function boolstr( val : boolean; cnt : byte ) : string';
20841:   Function CompareByte( buf1, buf2 : pchar; len : longint ) : integer';
20842:   Function Trimx( const S : string ) : string';
20843:   Function Printf2( const s : string; var Buf: string; size : word ) : string';
20844:   Function Printfx( const s : string; var Buf: string; size : word ) : string';
20845:   Function FillTo( s : string; tolenth : integer ) : string';
20846:   Function stringdup( const s : string ) : string';
20847:   //Procedure stringdispose( var p : pShortstring );
20848:   Function EscapeToPascal( const s : string; var code : integer ) : string';
20849:   Function ValDecimal( const S : String; var code : integer ) : longint';

```

```

20850: Function ValUnsignedDecimal( const S : String; var code : integer ) : longword';
20851: Function ValOctal( const S : String; var code : integer ) : longint';
20852: Function ValBinary( const S : String; var code : integer ) : longint';
20853: Function ValHexadecimal( const S : String; var code : integer ) : longint';
20854: Function CleanString( const s : string ) : string';
20855: Function ChangeFileExt2( const FileName, Extension : string ) : string';
20856: Function fillwithzero( s : string; newlength : integer ) : string';
20857: Function removenulls( const s : string ) : string';
20858: //CL.AddConstantN('WhiteSpace','Char').SetString(' ' or #10 or #13 or #9);
20859: end;
20860:
20861: procedure SIRegister_iertf(CL: TPSPascalCompiler);
20862: begin
20863:   Function mime_isvalidcontenttype( const s : shortstring ) : boolean';
20864:   Function langtag_isvalid( const s : string ) : boolean';
20865:   Function langtag_split( const s : string; var primary, sub : string ) : boolean';
20866:   URI_START_DELIMITER_CHAR,'String').SetString('<');
20867:   URI_END_DELIMITER_CHAR,'String').SetString('>');
20868:   URI_SCHEME_NAME_EMAIL,'String').SetString('mailto');
20869:   URI_SCHEME_SEPARATOR,'String').SetString(':');
20870:   Function uri_split( url : string; var scheme, authority, path, query : string ) : boolean';
20871:   Function urn_isvalid( s : shortstring ) : boolean';
20872:   Function urn_isvalidnid( nid : string ) : boolean';
20873:   Function urn_split( urn : string; var urnidstr, nidstr, nssstr : string ) : boolean';
20874:   Function urn_pathsplit( path : string; var namespace, nss : string ) : boolean';
20875:   Function http_pathsplit( path : string; var directory, name : string ) : boolean';
20876:   Function file_pathsplit( path : string; var directory, name : string ) : boolean';
20877: end;
20878:
20879: procedure SIRegister_dateutilreal(CL: TPSPascalCompiler);
20880: begin
20881:   CL.AddTypeS('TDatetimeReal', 'real');
20882:   CL.AddTypeS('TDateInfo', 'record DateTime : TDateTime; UTC : boolean; end');
20883:   //CL.AddTypeS('float', 'real');
20884:   CL.AddTypeS('big_integer_t', 'int64');
20885:   'DayMondayR','LongInt').SetInt( 1 );
20886:   CL.AddConstantN('DayTuesdayR','LongInt').SetInt( 2 );
20887:   'DayWednesdayR','LongInt').SetInt( 3 );
20888:   'DayThursdayR','LongInt').SetInt( 4 );
20889:   'DayFridayR','LongInt').SetInt( 5 );
20890:   'DaySaturdayR','LongInt').SetInt( 6 );
20891:   'DaySundayR','LongInt').SetInt( 7 );
20892:   Function CurrentYearreal : word');
20893:   Function Datereal : TDatetimeReal');
20894:   Function Dateofreal( const AValue : TDatetimeReal ) : TDatetimeReal');
20895:   Function DateTimeToStrreal( DateTime : TDatetimeReal ) : string');
20896:   Function DateToStrreal( date : TDatetimeReal ) : string');
20897:   Function Dayofreal( const AValue : TDatetimeReal ) : Word');
20898:   Function DaysBetweenreal( const ANow, AThen : TDatetimeReal ) : integer');
20899:   Procedure DecodeDatereal( Date : TDatetimeReal; var Year, Month, Day : Word );
20900:   Procedure DecodeDateTimereal( const AValue : TDatetimeReal; var Year, Month, Day, Hour, Minute, Second, Millisecond : Word );
20901:   Procedure DecodeTimereal( Time : TDatetimeReal; var Hour, Min, Sec, MSec : Word );
20902:   Function HourOfreal( const AValue : TDatetimeReal ) : Word';
20903:   Function IncDayreal( const AValue : TDatetimeReal; const ANumberOfDays : Integer ) : TDatetimeReal';
20904:   Function IncHourreal( const AValue:TDatetimeReal; const ANumberOfHours:longint):TDatetimeReal';
20905:   Function IncMillisecondreal( const AValue : TDatetimeReal; const ANumberOfMilliseconds : big_integer_t ) : TDatetimeReal );
20906:   Function IncMinutereal( const AValue : TDatetimeReal; const ANumberOfMinutes : big_integer_t ) : TDatetimeReal );
20907:   Function IncSecondreal( const AValue : TDatetimeReal; const ANumberOfSeconds : big_integer_t ) : TDatetimeReal );
20908:   Function IncWeekreal( const AValue:TDatetimeReal;const ANumberOfWeeks:Integer): TDatetimeReal );
20909:   Function IsPMreal( const AValue : TDatetimeReal ) : Boolean';
20910:   Function IsValidDatereal( const AYear, AMonth, ADay : Word ) : Boolean';
20911:   Function IsValidDateTimereal( const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond : Word ) : Boolean';
20912:   Function IsValidTimereal( const AHour, AMinute, ASecond, AMilliSecond : Word ) : Boolean';
20913:   Function MinuteOfreal( const AValue : TDatetimeReal ) : Word';
20914:   Function MonthOfreal( const AValue : TDatetimeReal ) : Word';
20915:   Function Nowreal : TDatetimeReal';
20916:   Function SameDatereal( const A, B : TDatetimeReal ) : Boolean';
20917:   Function SameDateTimereal( const A, B : TDatetimeReal ) : Boolean';
20918:   Function SameTimereal( const A, B : TDatetimeReal ) : Boolean';
20919:   Function SecondOfreal( const AValue : TDatetimeReal ) : Word';
20920:   Function Timereal : TDatetimeReal';
20921:   Function Gettimereal : TDatetimeReal';
20922:   Function TimeOfreal( const AValue : TDatetimeReal ) : TDatetimeReal';
20923:   Function TimeToStrreal( Time : TDatetimeReal ) : string';
20924:   Function Todayreal : TDatetimeReal';
20925:   Function TryEncodeDatereal( Year, Month, Day : Word; var Date : TDatetimeReal ) : Boolean';
20926:   Function TryEncodeTimereal( Hour, Min, Sec, MSec : Word; var Time : TDatetimeReal ) : Boolean';
20927:   Function TryEncodeDateTimereal( const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond : Word; var AValue : TDatetimeReal ) : Boolean';
20928:   Function TryStrToDatereal( const S : string; var Value : TDatetimeReal ) : Boolean';
20929:   Function TryStrToDateTimereal( const S : string; var Value : TDatetimeReal ) : Boolean';
20930:   Function TryStrToTimereal( const S : string; var Value : TDatetimeReal ) : Boolean';
20931:   Function YearOfreal( const AValue : TDatetimeReal ) : Word');
20932:   Function parsetimeISO( timestr : string; var hourval, minval, secval : word; var offsetminval : integer; var UTC : boolean ) : boolean';

```

```

20933: Function parsedateISO( datestr : string; var yearval, monthval, dayval : word) : boolean';
20934: Function datetojd( year, month, day, hour, minute, second, millisecond : word) : float';
20935: Procedure jdtodate( jday : float; var year, month, day, hour, minute, second, msec : word')';
20936: Function converttoisotime( timestr : string) : string';
20937: Function AdobeDateToISODATE( s : string) : string';
20938: Function RFC822ToISODATETime( s : string) : string';
20939: Procedure getdatedatos( var year, month, mday, wday : word')';
20940: Procedure gettimedatos( var hour, minute, second, sec100 : word')';
20941: end;
20942:
20943: procedure SIRegister_dateext4(CL: TPSPascalCompiler);
20944: begin
20945:   CL.AddTypeS('tfiletime2', 'record LowDateTime: longword; HighDateTime: longword; end');
20946: // CL.AddTypeS('big_integer_t', 'int64');
20947:   Function DosToWinTime( DTime : longint; var Wtime : TFileTime) : longbool';
20948:   Function WinToDOSTime( const Wtime : TFileTime; var Dtime : longint) : longbool';
20949:   Function TryStrToDateExt(const S: string; var Value: TDateTime; var UTC:boolean): Boolean;
20950:   Function TryEncodeDateAndTimeToStr( const Year, Month, Day, Hour, Minute, Second, MilliSecond : word; UTC
: boolean; var AValue : string) : boolean';
20951:   Function DateToStringExt( DateTime : TDateTime; utc : boolean) : string';
20952:   Procedure GetCurrentDate( var Year, Month, Day, DayOfWeek : integer')';
20953:   Procedure GetcurrentTime2( var Hour, Minute, Second, Sec100 : integer')';
20954:   Function TryUNIXToDateExt( unixtime : big_integer_t; var DateTime : TDateTime; var UTC : boolean) :
boolean';
20955:   Function TryFileTimeToDateExt( ftime : tfiletime; var DateTime : TDateTime; var UTC : boolean) :
boolean';
20956:   procedure JulianToGregorian(JulianDN:big_integer_t;var Year,Month,Day:Word);';
20957: end;
20958:
20959:
20960: procedure SIRegister_locale(CL: TPSPascalCompiler);
20961: begin
20962:   Function GetISODateString( Year, Month, Day : Word) : string';
20963:   Function GetISODateStringBasic( Year, Month, Day : Word) : string';
20964:   Function IsValidISODateString( datestr : shortstring; strict : boolean) : boolean';
20965:   Function IsValidISODateStringExt(datestr:shortstring;strict:bool;var Year,Month,Day:word):bool;
20966:   Function IsValidISOTimeString( timestr : shortstring; strict : boolean) : boolean';
20967:   Function IsValidISOTimeStringExt( timestr : shortstring; strict : boolean; var hour, min, sec : word; var
offhour, offmin : smallint) : boolean';
20968:   Function IsValidISOTimeString( str : shortstring; strict : boolean) : boolean';
20969:   Function GetISOTimeString( Hour, Minute, Second : Word; UTC : Boolean) : shortstring';
20970:   Function GetISOTimeStringBasic(Hour,Minute,Second: Word; UTC : Boolean) : shortstring';
20971:   Function GetISODateTimeString(Year,Month,Day,Hour,Minute,Second:Word; UTC:Bool):shortstring';
20972:   Procedure UNIXToDateTime2(epoch:big_integer_t; var year,month,day,hour,minute,second: Word)';
20973:   Function GetCharEncoding( alias : string; var _name : string) : integer';
20974:   Function MicrosoftCodePageToMIMECharset( cp : word) : string';
20975:   Function MicrosoftLangageCodeToISOCode( langcode : integer) : string';
20976:   CHAR_ENCODING_UTF8', 'LongInt').SetInt( 0);
20977:   CL.AddConstantN('CHAR_ENCODING_UNKNOWN', 'LongInt').SetInt( - 1);
20978:   'CHAR_ENCODING_UTF32BE', 'LongInt').SetInt( 1);
20979:   'CHAR_ENCODING_UTF32LE', 'LongInt').SetInt( 2);
20980:   'CHAR_ENCODING_UTF16LE', 'LongInt').SetInt( 3);
20981:   'CHAR_ENCODING_UTF16BE', 'LongInt').SetInt( 4);
20982:   'CHAR_ENCODING_BYTE', 'LongInt').SetInt( 5);
20983:   'CHAR_ENCODING_UTF16', 'LongInt').SetInt( 6);
20984:   'CHAR_ENCODING_UTF32', 'LongInt').SetInt( 7);
20985: end;
20986:
20987: procedure SIRegister.Strings(CL: TPSPascalCompiler);
20988: begin
20989:   CL.AddDelphiFunction('Function StrLenPchar( Str : PChar) : longint');
20990:   Function StrEndPchar( Str : PChar) : PChar';
20991:   Function StrMovePchar( Dest, Source : Pchar; l : Longint) : pchar';
20992:   Function StrCopyPchar( Dest, Source : PChar) : PChar';
20993:   Function StrECopyPchar( Dest, Source : PChar) : PChar';
20994:   Function StrLCopyPchar( Dest, Source : PChar; MaxLen : Longint) : PChar';
20995:   Function StrPCopyPchar( Dest : PChar; Source : String) : PChar';
20996:   Function StrCatPchar( Dest, Source : PChar) : PChar';
20997:   Function strlcatPchar( dest, source : pchar; l : Longint) : pchar';
20998:   Function StrCompPchar( Str1, Str2 : PChar) : Integer';
20999:   Function StrICmpPchar( Str1, Str2 : PChar) : Integer';
21000:   Function StrLCompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Integer';
21001:   Function StrLICompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Integer';
21002:   Function StrScanPchar( Str : PChar; Ch : Char) : PChar';
21003:   Function StrRScanPchar( Str : PChar; Ch : Char) : PChar';
21004:   Function StrPosPchar( Str1, Str2 : PChar) : PChar';
21005:   Function StrUpperPchar( Str : PChar) : PChar';
21006:   Function StrLowerPchar( Str : PChar) : PChar';
21007:   Function StrPasPchar( Str : PChar) : String';
21008:   Function StrNewPchar( P : PChar) : PChar';
21009:   Procedure StrDisposePchar( P : PChar)';
21010: end;
21011:
21012: procedure SIRegister_crc_checks(CL: TPSPascalCompiler);
21013: begin
21014:   Function UpdateCrc32_2( InitCrc : longword; b : byte) : longword';
21015:   Function UpdateCrc16_2( InitCrc : word; b : byte) : word';
21016:   Function UpdateAdler32( InitAdler : longword; b : byte) : longword';
21017:   Function UpdateFletcher8( InitFletcher : word; b : byte) : word';

```

```

21018: Function UpdateCRC( InitCrc : word; b : byte) : word');
21019: end;
21020:
21021: Procedure SIRegister_extdos(CL: TPSPascalCompiler);
21022: begin
21023:   CL.AddTypeS('utf8char', 'char');
21024:   CL.AddTypeS('putf8char', 'pchar');
21025:   CL.AddTypeS('utf16char', 'word');
21026:   CL.AddTypeS('ucs4char', 'longword');
21027: //CL.AddTypeS('pucs4char', '^ucs4char // will not work');
21028: CL.AddTypeS('ucs2char', 'word');
21029: CL.AddTypeS('pucs2char', 'ucs2char'); // will not work';
21030: //CL.AddTypeS('pucs2char', '^ucs2char // will not work');
21031: CL.AddTypeS('utf8string', 'string');
21032: // CL.AddTypeS('putf8shortstring', '^shortstring // will not work');
21033: CL.AddTypeS('USER_INFO_2', 'record usri2_name : pucs2char; usri2_password : '
21034: +'pucs2char; usri2_password_age : DWORD; usri2_priv : DWORD; usri2_home_dir '
21035: +' : pucs2char; usri2_comment : pucs2char; usri2_flags : DWORD; usri2_script '
21036: +'path : pucs2char; usri2_auth_flags : DWORD; usri2_full_name : pucs2char; u'
21037: +'sri2_usr_comment : pucs2char; usri2_parms : pucs2char; usri2_workstations '
21038: +' : pucs2char; usri2_last_logon : DWORD; usri2_last_logoff : DWORD; usri2_ac'
21039: +'ctExpires : DWORD; usri2_max_storage : DWORD; usri2_units_per_week : DWOR'
21040: +'D; usri2_logon_hours : pchar; usri2_bad_pw_count : DWORD; usri2_num_logons'
21041: +' : DWORD; usri2_logon_server : pucs2char; usri2_country_code : DWORD; usri2_code_page : DWORD; end');
21042: //CL.AddTypeS('P_USER_INFO_2', '^USER_INFO_2 // will not work');
21043: CL.AddTypeS('SE_OBJECT_TYPE', '( SE_UNKNOWN_OBJECT_TYPE, SE_FILE_OBJECT, SE_S'
21044: +'ERVICE, SE_PRINTER, SE_REGISTRY_KEY, SE_IMSHARE, SE_KERNEL_OBJECT, SE_WIND'
21045: +'OW_OBJECT, SE_DS_OBJECT, SE_DS_OBJECT_ALL, SE_PROVIDER_DEFINED_OBJECT, SE_'
21046: +'WMIGUID_OBJECT )');
21047: //CL.AddTypeS('PPSID', '^PPSID // will not work');
21048: CL.AddTypeS('ASSOCF', 'DWORD');
21049: CL.AddTypeS('ASSOCSTR', '( ASSOCSTR_NONE, ASSOCSTR_COMMAND, ASSOCSTR_EXECUTAB'
21050: +'LE, ASSOCSTR_FRIENDLYDOCNAME, ASSOCSTR_FRIENDLYAPPNAME, ASSOCSTR_NOOPEN, A'
21051: +'SSOCSTR_SHELLNEWVALUE, ASSOCSTR_DDECOMMAND, ASSOCSTR_DDEIFEXEC, ASSOCSTR_DDEAPPLICATION,
ASSOCSTR_DDETOPIC )');
21052: CL.AddTypeS('TCHAR', 'ucs2char');
21053: (CL.AddTypeS('TFileTime', 'FILETIME');
21054: CL.AddTypeS('TWin32FindDataW', '_WIN32_FIND_DATAW');
21055: CL.AddTypeS('TCHAR', 'ucs2char');
21056: CL.AddTypeS('TFileTime', 'FILETIME');
21057: CL.AddTypeS('TWin32FindDataW', '_WIN32_FIND_DATAW'); )
21058: //dev: array[0..127] of char;
21059: /* Unique file serial number, this may change from one boot to the next. */
21060: //ino: array[0..127] of char;
21061:
21062: CL.AddTypeS('treourceattribute', '( attr_any, attr_READONLY, attr_HIDDEN, at'
21063: +'tr_system, attr_ARCHIVE, attr_LINK, attr_DIRECTORY, attr_TEMPORARY, attr_E'
21064: +'ncrypted, attr_NO_INDEXING, attr_EXTENDED, attr_COMPRESSED, attr_OFFLINE, attr_SPARSE
) );
21065: TFileAssociation', 'record appname : utf8string; exename : utf8string; end');
21066: tresourceattributes', 'set of treourceattribute');
21067: CL.AddTypeS('TFileStats', 'record name : utf8string; size : big_integer_t; ow'
21068: +'ner : utf8string; ctime : TDateTime; mtime : TDateTime; atime : TDateTime; '
21069: +'nlink : integer; attributes : tresourceattributes; association : tfileass'
21070: +'ociation; streamcount: integer; accesses: integer; utc: boolean; dev: array[0..127] of char; '
21071: +'ino: array[0..127] of char; comment : utf8string; dirstr : utf8string; end');
21072: TSearchRecExt', 'record Stats : TFileStats; FindHandle : THandle'
21073: +'; W32FindData : TWin32FindData; IncludeAttr : longint; SearchAttr : TResourceAttributes; end');
21074: function GetFileOwner( fname : putf8char) : utf8string';
21075: GetFileAtime( fname : putf8char; var atime : TDateTime) : integer';
21076: GetFileMTime( fname : putf8char; var mtime : TDateTime) : integer';
21077: GetFileCTime( fname : putf8char; var ctime : TDateTime) : integer';
21078: GetFileSizeExt( fname : putf8char) : big_integer_t';
21079: GetFileAttributesExt( fname : putf8char) : tresourceattributes';
21080: GetFilestats( fname : putf8char; var stats : TFileStats) : integer';
21081: DirectoryExistsExt( DName : utf8string) : Boolean';
21082: FileExistsExt( const FName : utf8string) : Boolean';
21083: GetCurrentDirectoryExt( var DirStr : utf8string) : boolean';
21084: SetCurrentDirectoryExt( const DirStr : utf8string) : boolean';
21085: SetFileATime( fname : putf8char; newatime : tdatetime) : integer';
21086: SetFileMTime( fname : putf8char; newmtime : tdatetime) : integer';
21087: SetFileCTime( fname : putf8char; newctime : tdatetime) : integer';
21088: FindFirstEx(path:putf8char; attr:tresourceattributes;var SearchRec:TSearchRecExt): int;
21089: FindNextEx( var SearchRec : TSearchRecExt) : integer';
21090: Procedure FindCloseEx( var SearchRec : TSearchRecExt)';
21091: GetUserFullName( account : utf8string) : utf8string';
21092: GetLoginConfigDirectory : utf8string';
21093: GetGlobalConfigDirectory : utf8string';
21094: GetLoginHomeDirectory : utf8string';
21095: // ucs4strnewstr(str: string; srctype: string): pucs4char;');
21096: // ucs4strnewucs4(src: pucs4char): pucs4char;');
21097: end;
21098:
21099:
21100: function TRestRequest_createStringStreamFromStringList(strings:TStringList):TStringStream;
21101:
21102: {A simple Oscilloscope using TWaveIn class.
21103: More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
21104: http://www.retroarchive.org/garbo/pc/turbopas/index.html

```

```

21105: uses
21106:   Forms,
21107:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
21108:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
21109:   uColorFunctions in 'uColorFunctions.pas',
21110:   AMixer in 'AMixer.pas',
21111:   uSettings in 'uSettings.pas',
21112:   UWavein4 in 'UWavein4.pas',
21113:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
21114:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame},
21115:   BSpectrum in unit uPSI_BSpectrum; {SpectralLibrary}
21116:   1250 unit uPSI_U_Oscilloscope4_2
21117:
21118: Functions_max hex in the box maxbox
21119: functionslist.txt
21120: FunctionsList1 3.9.9./88/91/92/94/95/96/98/100/101/110/120/160/180/190/192/195/V402/V420/V422/V424.60.80
21121:
21122: ****
21123: Procedure
21124: PROCEDURE SIZE 10017 9475 8875 8396 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
21125: Procedure *****Now the Procedure list*****
21126: Procedure ( ACol, ARow : Integer; Items : TStrings)
21127: Procedure ( Agg : TAggregate)
21128: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
21129: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
21130: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
21131: Procedure ( ASender : TObject; const ABytes : Integer)
21132: Procedure ( ASender : TObject; VStream : TStream)
21133: Procedure ( AThread : TIdThread)
21134: Procedure ( AWebModule : TComponent)
21135: Procedure ( Column : TColumn)
21136: Procedure (const AUsername: String; const APASSWORD : String; AAAuthenticationResult : Boolean)
21137: Procedure ( const iStart : integer; const sText : string)
21138: Procedure (Control:TCustomTabControl; TabIndex:Integer; const Rect : TRect; Active : Boolean)
21139: Procedure ( Database : TDatabase; LoginParams : TStrings)
21140: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
Action:TReconcileAction)
21141: Procedure ( DATASET : TDATASET)
21142: Procedure ( DataSet:TDataSet;E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
21143: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
21144: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
21145: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
21146: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
21147: Procedure ( Done : Integer)
21148: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
21149: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection;const Rect:TRect;Pressed:Bool)
21150: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
21151: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
21152: Procedure (HeaderControl:THeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Bool)
21153: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
21154: Procedure (Sender:TCustomListView;const ARect:TRect;Stage:TCustomDrawStage;var DefaultDraw: Bool)
21155: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
21156: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
21157: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
21158: Procedure ( SENDER : TFIELD; const TEXT : String)
21159: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
21160: Procedure ( Sender : TIdTelnet; const Buffer : String)
21161: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
21162: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
21163: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
21164: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
21165: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
21166: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
21167: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
21168: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
21169: Procedure ( Sender : TObject; Button : TMPBnType)
21170: Procedure ( Sender : TObject; Button : TMPBnType; var DoDefault : Boolean)
21171: Procedure ( Sender : TObject; Button : TUDBnType)
21172: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect: TRect; var DoneDrawing : Boolean)
21173: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
21174: Procedure ( Sender : TObject; Column : TListColumn)
21175: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
21176: Procedure ( Sender : TObject; Connecting : Boolean)
21177: Procedure (Sender:TObject; const PapSize:SmallInt; const Orient:TPrtOrient; const PageTy:TPageTy;var
DoneDraw:Bool
21178: Procedure (Sender:TObject; const Rect:TRect; DataCol:Int; Column:TColumn; State:TGridDrawState)
21179: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
21180: Procedure (Sender:TObject; const UserStr:string; var DateAndTime:TDateTime; var AllowChange:Bool)
21181: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
21182: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
21183: Procedure ( Sender : TObject; FromSection,ToSection: THeaderSection; var AllowDrag : Boolean)
21184: Procedure ( Sender : TObject; Index : LongInt)
21185: Procedure ( Sender : TObject; Item : TListItem)
21186: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
21187: Procedure ( Sender : TObject; Item : TListItem; Change: TItemChange; var AllowChange: Boolean)
21188: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
21189: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
21190: Procedure ( Sender : TObject; Item : TListItem; var S : string)
21191: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)

```

```

21192: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
21193: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
21194: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
21195: Procedure ( Sender : TObject; Node : TTreenode)
21196: Procedure ( Sender : TObject; Node : TTreenode; var AllowChange : Boolean)
21197: Procedure ( Sender : TObject; Node : TTreenode; var AllowCollapse : Boolean)
21198: Procedure ( Sender : TObject; Node : TTreenode; var AllowEdit : Boolean)
21199: Procedure ( Sender : TObject; Node : TTreenode; var AllowExpansion : Boolean)
21200: Procedure ( Sender : TObject; Node : TTreenode; var S : string)
21201: Procedure ( Sender : TObject; Node1, Node2 : TTreenode; Data : Integer; var Compare : Integer)
21202: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
21203: Procedure ( Sender : TObject; Rect : TRect)
21204: Procedure (Sender:TObject; Request:TWebRequest; Response:TWebResponse; var Handled : Boolean)
21205: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Int;Orient:TPageScrollerOrientation;var Delta:Int)
21206: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
21207: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent:TErrorEvent;var ErrorCode : Int)
21208: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
21209: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
21210: Procedure ( SENDER : TOBJECT; SOURCE : TMENUTITEM; REBUILD : BOOLEAN)
21211: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
21212: Procedure ( Sender : TObject; TabCanvas: TCanvas; R:TRect; Index: Integer; Selected : Boolean)
21213: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
21214: Procedure ( Sender : TObject; Thread : TServerClientThread)
21215: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
21216: Procedure ( Sender : TObject; Username, Password : string)
21217: Procedure ( Sender : TObject; var AllowChange : Boolean)
21218: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
21219: Procedure ( Sender : TObject; var Caption : string; var Alignment : THMLCaptionAlignment)
21220: Procedure ( Sender : TObject; var Continue : Boolean)
21221: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
21222: Procedure ( Sender : TObject; var Username : string)
21223: Procedure ( Sender : TObject; Wnd : HWND)
21224: Procedure ( Sender : TToolbar; Button : TToolbutton)
21225: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Bool)
21226: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
21227: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
21228: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolbutton)
21229: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
21230: Procedure ( StatusBar : TStatusbar; Panel : TStatusPanel; const Rect : TRect)
21231: Procedure (var FieldNames:TWideStrings;SQL:WideString;var BindAllFields:Bool;var TableName : WideString)
21232: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
21233: procedure (Sender: TObject)
21234: procedure (Sender: TObject; var Done: Boolean)
21235: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
21236: procedure _T(Name: tbtString; v: Variant);
21237: Procedure AbandonSignalHandler( RtlSigNum : Integer)
21238: Procedure Abort
21239: Procedure About1Click( Sender : TObject)
21240: Procedure Accept( Socket : TSocket)
21241: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
21242: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
21243: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
21244: Procedure AESEncryptString(const plaintext: string; var ciphertext:string; password: string)
21245: Procedure AESDecryptString(var plaintext: string; const ciphertext:string; password: string)
21246: Procedure Add( Addend1, Addend2 : TMyBigInt)
21247: Procedure ADD( const AKEY, AVALUE : VARIANT)
21248: Procedure Add( const Key : string; Value : Integer)
21249: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
21250: Procedure ADD( FIELD : TFIELD)
21251: Procedure ADD( ITEM : TMENUTITEM)
21252: Procedure ADD( POPUP : TPOPUPMENU)
21253: Procedure AddCharacters( xCharacters : TCharSet)
21254: Procedure AddDriver( const Name : string; List : TStrings)
21255: Procedure AddImages( Value : TCustomImageList)
21256: Procedure AddIndex(const Name,Fields:string; Options:TIndexOptions;const DescFields: string)
21257: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
21258: Procedure AddLoader( Loader : TBitmapLoader)
21259: Procedure ADDPARAM( VALUE : TPARAM)
21260: Procedure AddPassword( const Password : string)
21261: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
21262: Procedure AddState( oState : TniRegularExpressionState)
21263: Procedure AddStrings( Strings : TStrings);
21264: procedure AddStrings(Strings: TStrings);
21265: Procedure AddStrings1( Strings : TWideStrings);
21266: Procedure AddStringTerm( var sString : string; const sTerm: string; const sSeparator : string)
21267: Procedure AddToRecentDocs( const Filename : string)
21268: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
21269: Procedure AllFunctionsList1Click(Sender: TObject)
21270: procedure AllObjectsList1Click(Sender: TObject);
21271: Procedure Allocate( AAllocateBytes : Integer)
21272: procedure AllResourceList1Click(Sender: TObject);
21273: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
21274: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
21275: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
21276: Procedure AnsiFree( var s : AnsiString)
21277: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
21278: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
21279: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
21280: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)

```

```

21281: Procedure AntiFreeze;
21282: Procedure APPEND
21283: Procedure Append( const S : WideString)
21284: procedure Append(S: string);
21285: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
21286: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
21287: Procedure AppendChunk( Val : OleVariant)
21288: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
21289: Procedure AppendStr( var Dest : string; S : string)
21290: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALen : Integer)
21291: Procedure ApplyRange
21292: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
21293: Procedure Arrange( Code : TListArrangement)
21294: procedure Assert(expr : Boolean; const msg: string);
21295: procedure Assert2(expr : Boolean; const msg: string);
21296: Procedure Assign( Alist : TCustomBucketList)
21297: Procedure Assign( Other : TObject)
21298: Procedure Assign( Source : TDragObject)
21299: Procedure Assign( Source : TPersistent)
21300: Procedure Assign(Source: TPersistent)
21301: procedure Assign2(mystring, mypath: string);
21302: Procedure AssignCurValues( Source : TDataSet);
21303: Procedure AssignCurValues1( const CurValues : Variant);
21304: Procedure ASSIGNFIELD( FIELD : TFIELD)
21305: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
21306: Procedure AssignFile(var F: Text; FileName: string)
21307: procedure AssignFile(var F: TextFile; FileName: string)
21308: procedure AssignFileRead(var mystring, myfilename: string);
21309: procedure AssignFileWrite(mystring, myfilename: string);
21310: Procedure AssignTo( Other : TObject)
21311: Procedure AssignValues( Value : TParameters)
21312: Procedure ASSIGNVALUES( VALUE : TPARAMS)
21313: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
21314: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
21315: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
21316: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
21317: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
21318: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
21319: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
21320: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
21321: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
21322: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
21323: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
21324: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
21325: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
21326: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
21327: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
21328: procedure Beep
21329: Procedure BeepOk
21330: Procedure BeepQuestion
21331: Procedure BeepHand
21332: Procedure BeepExclamation
21333: Procedure BeepAsterisk
21334: Procedure BeepInformation
21335: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
21336: Procedure BeginLayout
21337: Procedure BeginTimer( const Delay, Resolution : Cardinal)
21338: Procedure BeginUpdate
21339: procedure BeginUpdate;
21340: procedure BigScreen1Click(Sender: TObject);
21341: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
21342: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
21343: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
21344: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
21345: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
21346: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
21347: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
21348: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
21349: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
21350: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
21351: Procedure BreakPointMenuClick( Sender : TObject)
21352: procedure BRINGTOFRONT
21353: procedure BringToFront;
21354: Procedure btnBackClick( Sender : TObject)
21355: Procedure btnBrowseClick( Sender : TObject)
21356: Procedure BtnClick( Index : TNavigateBtn)
21357: Procedure btnLargeIconsClick( Sender : TObject)
21358: Procedure BuildAndSendRequest( AURI : TIdURI)
21359: Procedure BuildCache
21360: Procedure BurnMemory( var Buff, BuffLen : integer)
21361: Procedure BurnMemoryStream( Destructo : TMemoryStream)
21362: Procedure CalculateFirstSet
21363: Procedure Cancel
21364: procedure CancelDrag;
21365: Procedure CancelEdit
21366: procedure CANCELHINT
21367: Procedure CancelRange
21368: Procedure CancelUpdates
21369: Procedure CancelWriteBuffer

```

```

21370: Procedure Capture1 (ADest:TStream;out VLineCount:Int;const ADelim:string;const AIsRFCMessage:Bool;
21371: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
21372: Procedure Capture3 (ADest:TStrings;out VLineCount:Int;const ADelim:string;const AIsRFCMessage:Bool
21373: procedure CaptureScreenFormat(vname: string; vextension: string);
21374: procedure CaptureScreenPNG(vname: string);
21375: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
21376: procedure CASCADE
21377: Procedure CastNativeToSoap(Info:PTTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Bool)
21378: Procedure CastSoapToVariant(SoapInfo:PTTypeInfo; const SoapData:WideString; NatData:Pointer);
21379: Procedure cbPathClick( Sender : TObject)
21380: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
21381: Procedure cedebugAfterExecute( Sender : TPSScript)
21382: Procedure cedebugBreakpoint(Sender: TObject;const FileName:String;Position,Row,Col:Cardinal)
21383: Procedure cedebugCompile( Sender : TPSScript)
21384: Procedure cedebugExecute( Sender : TPSScript)
21385: Procedure cedebugIdle( Sender : TObject)
21386: Procedure cedebugLineInfo(Sender:TObject; const FileName: String; Position,Row,Col: Cardinal)
21387: Procedure CenterHeight( const pc, pcParent : TControl)
21388: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
21389: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
21390: Procedure Change
21391: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
21392: Procedure Changed
21393: Procedure ChangeDir( const ADirName : string)
21394: Procedure ChangeDirUp
21395: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
21396: Procedure ChangeLevelBy( Value : TChangeRange)
21397: Procedure ChDir(const s: string)
21398: Procedure Check(Status: Integer)
21399: Procedure CheckCommonControl( CC : Integer)
21400: Procedure CHECKFIELDNAME( const FIELDNAME : String)
21401: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
21402: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected:bool;const AIgnoreBuffer:bool)
21403: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
21404: Procedure CheckToken( T : Char)
21405: procedure CheckToken(t:char)
21406: Procedure CheckTokenSymbol( const S : string)
21407: procedure CheckTokenSymbol(s:string)
21408: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
21409: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
21410: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
21411: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
21412: procedure CipherFile1Click(Sender: TObject);
21413: Procedure Clear;
21414: Procedure Clear1Click( Sender : TObject)
21415: Procedure ClearColor( Color : TColor)
21416: Procedure CLEARITEM( AITEM : TMENUITEM)
21417: Procedure ClearMapping
21418: Procedure ClearSelection( KeepPrimary : Boolean)
21419: Procedure ClearWriteBuffer
21420: Procedure Click
21421: Procedure Close
21422: Procedure Close1Click( Sender : TObject)
21423: Procedure CloseDatabase( Database : TDatabase)
21424: Procedure CloseDataSets
21425: Procedure CloseDialog
21426: Procedure CloseFile(var F: Text);
21427: Procedure Closure
21428: Procedure CMYKToBGR(const Source,Target:Pointer; const BitsPerSample: Byte; Count : Cardinal);
21429: Procedure CMYKToBGR1(const C,M,Y,K, Target:Pointer;const BitsPerSample: Byte; Count:Cardinal);
21430: Procedure CodeCompletionList1Click( Sender : TObject)
21431: Procedure ColEnter
21432: Procedure Collapse
21433: Procedure Collapse( Recurse : Boolean)
21434: Procedure ColorRGBtoHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
21435: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
21436: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
21437: Procedure Compile1Click( Sender : TObject)
21438: procedure ComponentCount1Click(Sender: TObject);
21439: Procedure Compress(azipfolder, azipfile: string)
21440: Procedure DeCompress(azipfolder, azipfile: string)
21441: Procedure XZip(azipfolder, azipfile: string)
21442: Procedure XUnZip(azipfolder, azipfile: string)
21443: Procedure Connect( const ATimeout : Integer)
21444: Procedure Connect( Socket : TSocket)
21445: procedure Console1Click(Sender: TObject);
21446: Procedure Continue
21447: Procedure ContinueCount( var Counter : TJclCounter)
21448: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
21449: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
21450: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
21451: Procedure ConvertImage(vsource, vdestination: string);
21452: // Ex. ConvertImage (Exxpath+'my233.bmp',Exxpath+'mypng111.png')
21453: Procedure ConvertBitmap(vsource, vdestination: string);
21454: Procedure ConvertToGray(Cnv: TCanvas);
21455: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
21456: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
21457: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
21458: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)

```

```

21459: Procedure CopyBytesToHostWord(const ASource:TIdBytes;const ASourceIndex:Int;var VDest: Word)
21460: Procedure CopyFrom( mbCopy : TMyBigInt)
21461: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
21462: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
21463: Procedure CopyTidByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
   of Byte; const ADestIndex : Integer; const ALength : Integer)
21464: Procedure CopyTidBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
   ALen:Int)
21465: Procedure CopyTidCardinal (const ASource:Cardinal; var VDest:TIdBytes;const ADestIndex:Integer)
21466: Procedure CopyTidInt64 (const ASource: Int64; var VDest : TIdBytes; const ADestIndex : Integer)
21467: Procedure CopyTidIPv6Address (const ASource:TIdIPv6Address; var VDest:TIdBytes;const ADestIndex:Integer)
21468: Procedure CopyTidLongWord(const ASource:LongWord;var VDest:TIdBytes;const ADestIndex : Integer)
21469: Procedure CopyTidNetworkLongWord( const ASource: LongWord;var VDest:TIdBytes;const ADestIndex:Integer)
21470: Procedure CopyTidNetworkWord(const ASource:Word; var VDest:TIdBytes;const ADestIndex : Integer)
21471: Procedure CopyTidString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
21472: Procedure CopyTidWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
21473: Procedure CopyToClipboard
21474: Procedure CountParts
21475: Procedure CreateDataSet
21476: Procedure CreateEmptyFile( const FileName : string)
21477: Procedure CreateFileFromString( const FileName, Data : string)
21478: Procedure CreateFromDelta( Source : TPacketDataSet)
21479: procedure CREATEHANDLE
21480: Procedure CreatePipeStreams (var InPipe:TInputPipeStream;var
   OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
21481: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
21482: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environ:PChar)
21483: Procedure CreateTable
21484: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
21485: procedure CSyntax1Click(Sender: TObject);
21486: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
21487: Procedure CURSORPOSCHANGED
21488: procedure CutFirstDirectory(var S: String)
21489: Procedure DataBaseError (const Message: string)
21490: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
21491: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
21492: Procedure DateTimeToSystemTime( DateTime: TDateTime; var SystemTime : TSystemTime)
21493: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
21494: Procedure DBIError(errorCode: Integer)
21495: Procedure DebugOutput( const AText: string)
21496: procedure DebugIn(DebugLOGFILE: string; Event_message: string);
21497: Procedure DebugRun1Click( Sender : TObject)
21498: procedure Dec;
21499: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
21500: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
21501: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
21502: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear,AMonth,AWeekOfMonth,ADayOfWeek :Word)
21503: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
21504: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
21505: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
21506: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
21507: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
21508: Procedure Decompile1Click( Sender : TObject)
21509: Procedure DefaultDrawColumnCell(const Rect:TRect;DataCol:Integer;Column:TColumn;State:TGridDrawState)
21510: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
21511: Procedure DeferLayout
21512: Procedure defFileRead
21513: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL; REMOVING:BOOLEAN)
21514: Procedure DelayMicroseconds( const MicroSeconds : Integer)
21515: Procedure Delete
21516: Procedure Delete( const Afilename : string)
21517: Procedure Delete( const Index : Integer)
21518: Procedure DELETE( INDEX : INTEGER)
21519: Procedure Delete( Index : LongInt)
21520: Procedure Delete( Node : TTreeNode)
21521: procedure Delete(var s: AnyString; ifrom, icoount: Longint);
21522: Procedure DeleteAlias( const Name : string)
21523: Procedure DeleteDriver( const Name : string)
21524: Procedure DeleteIndex( const Name : string)
21525: Procedure DeleteKey( const Section, Ident : String)
21526: Procedure DeleteRecords
21527: Procedure DeleteRecords( AffectRecords : TAffectRecords)
21528: Procedure DeleteString( var pStr : String; const pDelStr : string)
21529: Procedure DeleteTable
21530: procedure DelphiSite1Click(Sender: TObject);
21531: Procedure Deselect
21532: Procedure Deselect( Node : TTreeNode)
21533: procedure DestroyComponents
21534: Procedure DestroyHandle
21535: Procedure Diff( var X : array of Double)
21536: procedure Diff(var X: array of Double);
21537: Procedure DirCreate( const DirectoryName : String' );
21538: procedure DISABLEALIGN
21539: Procedure DisableConstraints
21540: Procedure Disconnect
21541: Procedure Disconnect( Socket : TSocket)
21542: Procedure Dispose
21543: procedure Dispose(P: PChar)
21544: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)

```

```

21545: Procedure DoKey( Key : TDBCtrlGridKey)
21546: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
21547: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
21548: Procedure Dormant
21549: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
21550: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
21551: Procedure DoubleToComp( Value : Double; var Result : Comp)
21552: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
21553: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
21554: procedure Draw(X, Y: Integer; Graphic: TGraphic);
21555: Procedure Draw(Canvas:TCanvas;X,Y,
  Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TIImageType;Enabled:Boolean);
21556: Procedure DrawArrow(ACanvas:TCanvas; Direction:TScrollBirection;Location:TPoint;Size:Integer)
21557: Procedure DrawCheck(ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
21558: Procedure DrawChevron(ACanvas:TCanvas; Direction: TScrollDirection; Location: TPoint; Size : Integer)
21559: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
21560: procedure DrawFocusRect(const Rect: TRect);
21561: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
21562: Procedure DRAWMENUEITEM(MENUITEM: TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE: TOWNERDRAWSTATE)
21563: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
21564: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
  TDrawingStyle; AImageType : TIImageType; Enabled : Boolean);
21565: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
21566: Procedure DrawPolyLine(const Canvas:TCanvas;var Points : TPointArray; const ClipRect : TRect)
21567: Procedure DropConnections
21568: Procedure DropDown
21569: Procedure DumpDescription( oStrings : TStrings)
21570: Procedure DumpStateTable( oStrings : TStrings)
21571: Procedure EDIT
21572: Procedure EditButtonClick
21573: Procedure EditFontClick( Sender : TObject)
21574: procedure Ellipse(X1, Y1, X2, Y2: Integer);
21575: Procedure Ellipse1( const Rect : TRect);
21576: Procedure EMMS
21577: Procedure Encode( ADest : TStream)
21578: procedure ENDDRAG(DROP:BOOLEAN)
21579: Procedure EndEdit( Cancel : Boolean)
21580: Procedure EndTimer
21581: Procedure EndUpdate
21582: Procedure EraseSection( const Section : string)
21583: Procedure Error( const Ident : string)
21584: procedure Error(Ident:Integer)
21585: Procedure ErrorFmt( const Ident : string; const Args : array of const)
21586: Procedure ErrorStr( const Message : string)
21587: procedure ErrorStr(Message:String)
21588: Procedure Exchange( Index1, Index2 : Integer)
21589: procedure Exchange(Index1, Index2: Integer);
21590: Procedure Exec( FileName, Parameters, Directory : string)
21591: Procedure ExecProc
21592: Procedure ExecSQL( UpdateKind : TUpdateKind)
21593: Procedure Execute
21594: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
21595: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
21596: Procedure ExecuteCommand(executeFile, paramstring: string)
21597: Procedure ExecuteShell(executeFile, paramstring: string)
21598: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
21599: Procedure ExitThread(ExitCode: Integer); stdcall;
21600: Procedure ExitProcess(ExitCode: Integer); stdcall;
21601: Procedure Expand( AUserName : String; AResults : TStrings)
21602: Procedure Expand( Recurse : Boolean)
21603: Procedure ExportClipboard1Click( Sender : TObject)
21604: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
21605: Procedure ExtractContentFields( Strings : TStrings)
21606: Procedure ExtractCookieFields( Strings : TStrings)
21607: Procedure ExtractFields(Separators,WhiteSpace:TSysCharSet; Content:PChar; Strings : TStrings)
21608: Procedure ExtractHeaderFields(Separ,
 WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
21609: Procedure ExtractHTTPFields(Separators,WhiteSpace:
  TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
21610: Procedure ExtractQueryFields( Strings : TStrings)
21611: Procedure FastDegToGrad
21612: Procedure FastDegToRad
21613: Procedure FastGradToDeg
21614: Procedure FastGradToRad
21615: Procedure FastRadToDeg
21616: Procedure FastRadToGrad
21617: Procedure FileClose( Handle : Integer)
21618: Procedure FileClose(handle: integer)
21619: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs>ShowDirs,Multitasking:Bool)
21620: Procedure FileStructure( AStructure : TIdFTPDataStructure)
21621: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
21622: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
21623: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
21624: Procedure FillChar2(var X: PChar ; count: integer; value: char)
21625: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
21626: Procedure FillIPList
21627: procedure FillRect(const Rect: TRect);
21628: Procedure FillTStrings( AStrings : TStrings)
21629: Procedure FilterOnBookmarks( Bookmarks : array of const)

```

```

21630: procedure FinalizePackage (Module: HMODULE)
21631: procedure FindClose;
21632: procedure FindClose2 (var F: TSearchRec);
21633: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
21634: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
21635: Procedure FindNearest( const KeyValues : array of const)
21636: Procedure FinishContext
21637: Procedure FIRST
21638: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
21639: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
21640: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
21641: Procedure FlushSchemaCache( const TableName : string)
21642: procedureFmtStr (var Result: string; const Format: string; const Args: array of const)
21643: procedure FOCUSCONTROL (CONTROL:TWINCONTROL)
21644: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
21645: Procedure FormActivate( Sender : TObject)
21646: procedure FormatLn(const format: String; const args: array of const); //alias
21647: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
21648: Procedure FormCreate( Sender : TObject)
21649: Procedure FormDestroy( Sender : TObject)
21650: Procedure FormKeyPress( Sender : TObject; var Key : Char)
21651: procedure FormOutput1Click(Sender: TObject);
21652: Procedure FormToHtml( Form : TForm; Path : string)
21653: procedure FrameRect(const Rect: TRect);
21654: Procedure Frame3D(Canvas:TCanvas;var Rect:TRect;TopColor,BottomColor:TColor; Width : Integer)
21655: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
21656: Procedure Free( Buffer : TRecordBuffer)
21657: Procedure Free( Buffer : TValueBuffer)
21658: Procedure Free;
21659: Procedure FreeAndNil (var Obj:TObject)
21660: Procedure FreeImage
21661: procedure FreeMem(P: PChar; Size: Integer)
21662: Procedure FreeTreeData( Tree : TUpdateTree)
21663: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
21664: Procedure FullCollapse
21665: Procedure FullExpand
21666: Procedure GenerateDBP(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
21667: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
21668: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
21669: Procedure Get1( AURL : string; const AResponseContent : TStream);
21670: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
21671: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: bool; AResume: Bool);
21672: Procedure GetAliasNames( List : TStrings)
21673: Procedure GetAliasParams( const AliasName : string; List : TStrings)
21674: Procedure GetApplicationsRunning( Strings : TStrings)
21675: Procedure getBox(aURL, extension: string);
21676: Procedure GetCommandTypes( List : TWideStrings)
21677: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
21678: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
21679: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
21680: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
21681: Procedure GetDatabaseNames( List : TStrings)
21682: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
21683: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
21684: Procedure GetDir(d: byte; var s: string)
21685: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
21686: Procedure GetDriverNames( List : TStrings)
21687: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
21688: Procedure GetDriverParams( const DriverName : string; List : TStrings)
21689: Procedure GetEmails1Click( Sender : TObject)
21690: Procedure getEnvironmentInfo;
21691: Function getEnvironmentString: string;
21692: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
21693: Procedure GetFieldNames( const TableName : string; List : TStrings)
21694: Procedure GetFieldNames( const TableName : string; List : TStrings);
21695: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
21696: Procedure GETFIELDNAMES( LIST : TSTRINGS)
21697: Procedure GetFieldNames1( const TableName : string; List : TStrings);
21698: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
21699: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
21700: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
21701: Procedure GetfileAttributeList( const Items : TStrings; const Attr : Integer)
21702: Procedure GetfileAttributeListEx( const Items : TStrings; const Attr : Integer)
21703: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
21704: Procedure GetFormatSettings
21705: Procedure GetFromDIB( var DIB : TBitmapInfo)
21706: Procedure GetFromHDIb( HDIB : HBitmap)
21707: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
21708: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
21709: Procedure GetIcon( Index : Integer; Image : TIcon);
21710: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
21711: Procedure GetIndexInfo( IndexName : string)
21712: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
21713: Procedure GetIndexNames( List : TStrings)
21714: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
21715: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
21716: Procedure GetIndexNames4( const TableName : string; List : TStrings);
21717: Procedure GetInternalResponse
21718: Procedure GETITEMNAMES( LIST : TSTRINGS)

```

```

21719: procedure GetMem(P: PChar; Size: Integer)
21720: Procedure GETOLE2ACCELERORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
21721: procedure GetPackageDescription(ModuleName: PChar): string)
21722: Procedure GetPackageNames( List : TStrings);
21723: Procedure GetPackageNames1( List : TWideStrings);
21724: Procedure GetParamList( List : TList; const ParamNames : WideString)
21725: Procedure GetProcedureNames( List : TStrings);
21726: Procedure GetProcedureNames( List : TWideStrings);
21727: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
21728: Procedure GetProcedureNames1( List : TStrings);
21729: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
21730: Procedure GetProcedureNames3( List : TWideStrings);
21731: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
21732: Procedure GetProcedureNames5( const PackageName,SchemaName: WideString; List : TWideStrings);
21733: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
21734: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
21735: Procedure GetProcedureParams2(ProcedureName,PackageName,SchemaName: Widestring; List: TList);
21736: Procedure GetProviderNames( Names : TWideStrings);
21737: Procedure GetProviderNames( Proc : TGetStrProc)
21738: Procedure GetProviderNames1( Names : TStrings);
21739: procedure GetQrCode2(Width,Height:Word;Correct_Level:string;const Data:string; apath: string);
21740: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string); //no open image
21741: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
Data:string);format:string):TLinearBitmap;
21742: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
21743: Procedure GetSchemaNames( List : TStrings);
21744: Procedure GetSchemaNames1( List : TWideStrings);
21745: Procedure getScriptandRunAsk;
21746: Procedure getScriptandRun(ascript: string);
21747: Procedure getScript(ascript: string); //alias
21748: Procedure getWebScript(ascript: string); //alias
21749: Procedure GetSessionNames( List : TStrings)
21750: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
21751: Procedure GetStrings( List : TStrings)
21752: Procedure GetSystemTime; stdcall;
21753: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
21754: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
21755: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
21756: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
21757: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
21758: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
21759: Procedure GetTableNames2(List : TWideStrings; SchemaName: WideString; SystemTables : Boolean);
21760: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
21761: Procedure GetVisibleWindows( List : Tstrings)
21762: Procedure GoBegin
21763: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
21764: Procedure GotoCurrent( Table : TTable)
21765: procedure GotoEnd1Click(Sender: TObject);
21766: Procedure GotoNearest
21767: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection)
21768: Procedure HandleException( E : Exception; var Handled : Boolean)
21769: procedure HANDLEMESSAGE
21770: procedure HandleNeeded;
21771: Procedure Head( AURL : string)
21772: Procedure Help( var AHelpContents : TStringList; ACommand : String)
21773: Procedure HexToBinary( Stream : TStream)
21774: procedure HexToBinary(Stream:TStream)
21775: Procedure HideDragImage
21776: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
21777: Procedure HideTraybar
21778: Procedure HideWindowForSeconds(secs: integer); //3 seconds
21779: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
21780: Procedure HookOSExceptions
21781: Procedure HookSignal( Rt1SigNum : Integer)
21782: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
21783: Procedure HTMLSyntax1Click( Sender : TObject)
21784: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
21785: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSEexec; x : TPSRuntimeClassImporter)
21786: Procedure ImportfromClipboard1Click( Sender : TObject)
21787: Procedure ImportfromClipboard2Click( Sender : TObject)
21788: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
21789: procedure Incb(var x: byte);
21790: Procedure Include1Click( Sender : TObject)
21791: Procedure IncludeOFF; //preprocessing
21792: Procedure IncludeON;
21793: procedure Info1Click(Sender: TObject);
21794: Procedure InitAltRecBuffers( CheckModified : Boolean)
21795: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
21796: Procedure InitContext( WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
21797: Procedure InitData( ASource : TDataSet)
21798: Procedure InitDelta( ADelta : TPacketDataSet);
21799: Procedure InitDelta1( const ADelta : OleVariant);
21800: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
21801: Procedure Initialize
21802: procedure InitializePackage(Module: HMODULE)
21803: Procedure INITIACTION
21804: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char;'
21805: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)

```

```

21806: Procedure InitModule( AModule : TComponent)
21807: Procedure InitStdConvs
21808: Procedure InitTreeData( Tree : TUpdateTree)
21809: Procedure INSERT
21810: Procedure Insert( Index : Integer; AClass : TClass)
21811: Procedure Insert( Index : Integer; AComponent : TComponent)
21812: Procedure Insert( Index : Integer; AObject : TObject)
21813: Procedure Insert( Index : Integer; const S : WideString)
21814: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
21815: Procedure Insert(Index: Integer; const S: string);
21816: procedure Insert(Index: Integer; S: string);
21817: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
21818: procedure InsertComponent(AComponent:TComponent)
21819: procedure InsertControl(AControl: TControl);
21820: Procedure InsertIcon( Index : Integer; Image : TIcon)
21821: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
21822: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
21823: procedure InsertObject(Index:Integer;S:String;AOBJECT:TObject)
21824: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
21825: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
21826: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
21827: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
21828: Procedure InternalBeforeResolve( Tree : TUpdateTree)
21829: Procedure InvalidateModuleCache
21830: Procedure InvalidateTitles
21831: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
21832: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
21833: Procedure InvalidDateTimeError(const AYear,AMth,Aday,AMin,ASec,AMilSec:Word;const
  ABaseDate:TDateTime)
21834: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
21835: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
21836: procedure JavaSyntax1Click(Sender: TObject);
21837: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
21838: Procedure KillDataChannel
21839: Procedure Largefont1Click( Sender : TObject)
21840: Procedure LAST
21841: Procedure LaunchCpl( FileName : string)
21842: Procedure Launch( const AFile : string)
21843: Procedure LaunchFile( const AFile : string)
21844: Procedure LetFileList(FileList: TStringlist; apath: string);
21845: Procedure lineToNumber( xmemo : String; met : boolean)
21846: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
  DefaultDraw:Bool)
21847: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
  : TCustomDrawState; var DefaultDraw : Boolean)
21848: Procedure ListViewData( Sender : TObject; Item : TListItem)
21849: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
  : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
21850: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
21851: Procedure ListViewDblClick( Sender : TObject)
21852: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
21853: Procedure ListDLEExports(const FileName: string; List: TStrings);
21854: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
21855: procedure LoadBytecode1Click(Sender: TObject);
21856: procedure LoadFromFileFromResource(const FileName: string; ms: TMemoryStream);
21857: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
21858: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
21859: Procedure LoadFromFile( AFileName : string)
21860: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
21861: Procedure LoadFromFile( const FileName : string)
21862: Procedure LOADFROMFILE( const FILENAME : String; BLOTYPE : TBLOTYPE)
21863: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
21864: Procedure LoadFromFile( const FileName : WideString)
21865: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
21866: Procedure LoadFromFile(const AFileName: string)
21867: procedure LoadFromFile(fileName:string);
21868: procedure LoadFromFile(fileName:String)
21869: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
21870: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
21871: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
21872: Procedure LoadFromStream( const Stream : TStream)
21873: Procedure LoadFromStream( S : TStream)
21874: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
21875: Procedure LoadFromStream(Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
21876: Procedure LoadFromStream( Stream : TStream)
21877: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOTYPE : TBLOTYPE)
21878: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
21879: procedure LoadFromStream(Stream: TStream);
21880: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
21881: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
21882: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
21883: Procedure LoadLastFile1Click( Sender : TObject)
21884: { LoadIcoToImage loads two icons from resource named NameRes,
  into two image lists ALarge and ASmall}
21885:   into two image lists ALarge and ASmall)
21886: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
21887: Procedure LoadMemo
21888: Procedure LoadParamsFromIniFile( FFileName : WideString)
21889: Procedure Lock
21890: Procedure Login

```

```

21891: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
21892: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
21893: Procedure MakeCaseInsensitive
21894: Procedure MakeDeterministic( var bChanged : boolean)
21895: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
21896: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
21897: Procedure MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:string);
21898: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
21899: Procedure SetComplexSoundElements(freqedt,Phaseddt,AmpEdt,WaveGrp:integer);
21900: Procedure SetRectComplexFormatStr( const S : string)
21901: Procedure SetPolarComplexFormatStr( const S : string)
21902: Procedure AddComplexSoundObjectToList (newf,newp,newa,news:integer; freqlist: TStrings);
21903: Procedure MakeVisible
21904: Procedure MakeVisible( PartialOK : Boolean)
21905: Procedure ManualClick( Sender : TObject)
21906: Procedure MarkReachable
21907: Procedure maBox; //shows the exe version data in a win box
21908: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
21909: Procedure Memo1Change( Sender : TObject)
21910: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var Action:TSynReplaceAction)
21911: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Int;var Special:Boolean;var FG,BG:TColor)
21912: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
21913: procedure Memory1Click(Sender: TObject);
21914: Procedure MERGE( MENU : TMAINMENU)
21915: Procedure MergeChangeLog
21916: procedure MINIMIZE
21917: Procedure MinimizeMaxbox;
21918: procedure MyCopyFile(Namel,Name2:string);
21919: Procedure MkDir(const s: string)
21920: Procedure MakeDir(const s: string)');
21921: Procedure ChangeDir(const s: string)');
21922: Function makefile(const FileName: string): integer)';
21923: Procedure mnuPrintFont1Click( Sender : TObject)
21924: procedure ModalStarted
21925: Procedure Modified
21926: Procedure ModifyAlias( Name : string; List : TStrings)
21927: Procedure ModifyDriver( Name : string; List : TStrings)
21928: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4,Skew,Kurtosis:Extended)
21929: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
21930: Procedure Move( CurIndex, NewIndex : Integer)
21931: procedure Move(CurIndex, NewIndex: Integer);
21932: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
21933: Procedure MoveChars(const ASource:String;ASourceStart:int;var ADest:String;ADestStart,ALen:int)
21934: Procedure moveCube( o : TMyLabel)
21935: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
21936: procedure MoveTo(X, Y: Integer);
21937: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
21938: Procedure MovePoint(var x,y:Extended; const angle:Extended);
21939: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
21940: Procedure Multiplyl( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
21941: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
21942: Procedure mxButton(x,y,width,height,top,left,aHandle: integer);
21943: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
21944: procedure New(P: PChar)
21945: procedure New1Click(Sender: TObject);
21946: procedure NewInstance1Click(Sender: TObject);
21947: Procedure NEXT
21948: Procedure NextMonth
21949: Procedure Noop
21950: Procedure NormalizePath( var APath : string)
21951: procedure ObjectBinaryToText(Input, Output: TStream)
21952: procedure ObjectBinaryToText1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
21953: procedure ObjectResourceToText(Input, Output: TStream)
21954: procedure ObjectResourceToText1 (Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
21955: procedure ObjectTextToBinary1(Input, Output: TStream)
21956: procedure ObjectTextToBinary1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
21957: procedure ObjectTextToResource(Input, Output: TStream)
21958: procedure ObjectTextToResource1 (Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
21959: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
21960: Procedure Open( const UserID : WideString; const Password : WideString);
21961: Procedure Open;
21962: Procedure open1Click( Sender : TObject)
21963: Procedure OpenCdDrive
21964: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
21965: Procedure OpenCurrent
21966: Procedure OpenFile(vfilenamepath: string)
21967: Procedure OpenDirectory1Click( Sender : TObject)
21968: Procedure OpenDir(adir: string);
21969: Procedure OpenIndexFile( const IndexName : string)
21970: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const SchemaID:OleVariant;DataSet:TADODataSet)
21971: Procedure OpenWriteBuffer( const AThreshold : Integer)
21972: Procedure OptimizeMem
21973: Procedure Options1( AURL : string);
21974: Procedure OutputDebugString(lpOutputString : PChar)
21975: Procedure PackBuffer
21976: Procedure Paint
21977: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)

```

```

21978: Procedure PaintToTBitmap( Target : TBitmap)
21979: Procedure PaletteChanged
21980: Procedure ParentBiDiModeChanged
21981: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
21982: Procedure PasteFromClipboard;
21983: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
21984: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
21985: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
21986: Procedure PError( Text : string)
21987: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
21988: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Int;X4:Int;Y4:Int);
21989: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
21990: procedure playmp3( mpath: string);
21991: Procedure PlayMP31Click( Sender : TObject)
21992: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
21993: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
21994: procedure PolyBezier(const Points: array of TPoint);
21995: procedure PolyBezierTo(const Points: array of TPoint);
21996: procedure Polygon(const Points: array of TPoint);
21997: procedure Polyline(const Points: array of TPoint);
21998: Procedure Pop
21999: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU; GROUPS: array of INT; var WIDTHS : array of LONGINT)
22000: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
22001: Procedure POPUP( X, Y : INTEGER)
22002: Procedure PopupURL(URL : WideString);
22003: Procedure POST
22004: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
22005: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
22006: Procedure Post6(AURL:string;const ASource:TIdMultiPartFormDataStream;AResponseContent:TStream);
22007: Procedure PostUser( const Email, FirstName, LastName : WideString)
22008: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
22009: procedure Pred(X: int64);
22010: Procedure Prepare
22011: Procedure PrepareStatement
22012: Procedure PreProcessXML( AList : TStrings)
22013: Procedure PreventDestruction
22014: Procedure Print( const Caption : string)
22015: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
22016: procedure printf(const format: String; const args: array of const);
22017: Procedure PrintList(Value: TStringlist);
22018: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);
22019: //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
22020: Procedure Printtout1Click( Sender : TObject)
22021: Procedure ProcessHeaders
22022: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
22023: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
22024: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
22025: Procedure ProcessMessage2(AMsg: TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
22026: Procedure ProcessMessagesOFF; //application.processmessages
22027: Procedure ProcessMessagesON;
22028: Procedure ProcessPath(const EditText:string;var Drive:Char;var DirPart:string;var FilePart: string)
22029: Procedure ProcessPath1(const EditText:string;var Drive:Char;var DirPart:string;var FilePart:string);
22030: Procedure Proclist Size is: 3797 /1415
22031: Procedure procMessClick( Sender : TObject)
22032: Procedure PSScriptCompile( Sender : TPSScript)
22033: Procedure PSScriptExecute( Sender : TPSScript)
22034: Procedure PSScriptLine( Sender : TObject)
22035: Procedure Push( ABoundary : string)
22036: procedure PushItem(AItem: Pointer)
22037: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
22038: Procedure Put2( const ASourceFile: string; const ADestFile : string; const AAppend : boolean);
22039: procedure PutLinuxLines(const Value: string)
22040: Procedure Quit
22041: Procedure RaiseConversionError( const AText : string);
22042: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
22043: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
22044: procedure RaiseException(Ex: TIEException; Param: String);
22045: Procedure RaiseExceptionForLastCmdResult;
22046: procedure RaiseLastException;
22047: procedure RaiseException2;
22048: Procedure RaiseException3(const Msg: string);
22049: Procedure RaiseExcept(const Msg: string);
22050: Procedure RaiseLastOSError
22051: Procedure RaiseLastWin32;
22052: procedure RaiseLastWin32Error)
22053: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
22054: Procedure RandomFillStream( Stream : TMemoryStream)
22055: procedure randomize;
22056: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
22057: Procedure RCS
22058: Procedure Read( Socket : TSocket)
22059: procedure Readln1(var ast: string); //of inputquery
22060: Procedure ReadBlobData
22061: procedure ReadBuffer(Buffer:String;Count:LongInt)
22062: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2,readonly:= false;
22063: Procedure ReadSection( const Section : string; Strings : TStrings)
22064: Procedure ReadSections( Strings : TStrings)
22065: Procedure ReadSections( Strings : TStrings);
22066: Procedure ReadSections1( const Section : string; Strings : TStrings);

```

```

22067: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
22068: Procedure ReadStream( AStream:TStream;AByteCount:LongInt;const AReadUntilDisconnect: boolean)
22069: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
22070: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
22071: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
22072: Procedure Realign;
22073: procedure Rectangle(X1, Y1, X2, Y2: Integer);
22074: Procedure Rectangle1( const Rect : TRect);
22075: Procedure RectCopy( var Dest : TRect; const Source : TRect)
22076: Procedure RectFitToScreen( var R : TRect)
22077: Procedure RectGrow( var R : TRect; const Delta : Integer)
22078: Procedure RectGrowX( var R : TRect; const Delta : Integer)
22079: Procedure RectGrowY( var R : TRect; const Delta : Integer)
22080: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
22081: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
22082: Procedure RectNormalize( var R : TRect)
22083: // TFileCallbackProcedure = procedure(filename:string);
22084: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean,callback: TFileCallbackProcedure);
22085: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
22086: Procedure RedirectTransition(oOldState:TniRegularExpressionState;oNewState:TniRegularExpressionState)
22087: Procedure Refresh;
22088: Procedure RefreshData( Options : TFetchOptions)
22089: Procedure REFRESHLOOKUPLIST
22090: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
22091: Procedure RegExPathfinder2(Pathin,fileout,firstp, aregex, ext: string; asort, acopy: boolean);
22092: Procedure RegisterAuthenticationMethod(MethodName: String; AuthClass : TIAuthENTICATIONCLASS)
22093: Procedure RegisterChanges( Value : TChangeLink)
22094: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
22095: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
22096: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
22097: Procedure ReInitialize( ADelay : Cardinal)
22098: procedure RELEASE
22099: Procedure Remove( const AByteCount : integer)
22100: Procedure REMOVE( FIELD : TFIELD)
22101: Procedure REMOVE( ITEM : TMENUITEM)
22102: Procedure REMOVE( POPUP : TPOPUPMENU)
22103: Procedure RemoveAllPasswords
22104: procedure RemoveComponent(AComponent:TComponent)
22105: Procedure RemoveDir( const AdirName : string)
22106: Procedure RemoveLambdaTransitions( var bChanged : boolean)
22107: Procedure REMOVEPARAM( VALUE : TPARAM)
22108: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
22109: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
22110: Procedure Rename( const ASourceFile, ADestFile : string)
22111: Procedure Rename( const FileName : string; Reload : Boolean)
22112: Procedure RenameTable( const NewTableName : string)
22113: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
22114: Procedure Replacee1Click( Sender : TObject)
22115: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
22116: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
22117: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
22118: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
22119: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
22120: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
22121: Procedure Requery( Options : TExecuteOptions)
22122: Procedure Reset
22123: Procedure Reset1Click( Sender : TObject)
22124: Procedure ResizeCanvas( XSiz,YSiz,XPos,YPos : Integer; Color : TColor)
22125: procedure ResourceExplore1Click(Sender: TObject);
22126: Procedure RestoreContents
22127: Procedure RestoreDefaults
22128: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
22129: Procedure RetrieveHeaders
22130: Procedure RevertRecord
22131: Procedure RGBAToBGRA(const Source, Target: Pointer; const BitsPerSample:Byte;Count : Cardinal)
22132: Procedure RGBToBGR( const Source, Target:Pointer; const BitsPerSample:Byte; Count : Cardinal);
22133: Procedure RGBToGR1(const R,G,B,Target: Pointer; const BitsPerSample : Byte; Count : Cardinal);
22134: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
22135: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
22136: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
22137: Procedure RleCompress2( Stream : TStream)
22138: Procedure RleDecompress2( Stream : TStream)
22139: Procedure RmDir(const S: string)
22140: Procedure Rollback
22141: Procedure Rollback( TransDesc : TTransactionDesc)
22142: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
22143: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
22144: Procedure RollbackTrans
22145: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
22146: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
22147: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
22148: Procedure RunDl132Internal(Wnd : HWnd; const DLLName,FuncName,CmdLine : string; CmdShow : Int)
22149: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
22150: Procedure S_EBox( const AText : string)
22151: Procedure S_GetEncryptionKeys(DatTime1,DTime2:TDateTime;var StartKey:int;var MultKey:int;var AddKey:int)
22152: Procedure S_IBox( const AText : string)
22153: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
22154: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
22155: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)

```

```

22156: Procedure SampleVarianceAndMean
22157:   (const X : TDynFloatArray; var Variance, Mean : Float)
22158: Procedure Save2Click( Sender : TObject)
22159: Procedure Saveas3Click( Sender : TObject)
22160: Procedure Savebefore1Click( Sender : TObject)
22161: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
22162: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
22163: Procedure SaveConfigFile
22164: Procedure SaveOutput1Click( Sender : TObject)
22165: procedure SaveScreenshotClick(Sender: TObject);
22166: Procedure SaveLn(pathname,content: string); //SaveIn(exepath+'mysavelntest.txt', memo2.text);
22167: Procedure SaveToClipboardFormat(var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
22168: Procedure SaveToClipboardFormat(var Format: Word; var Data : THandle; var APalette : HPALETTE)
22169: Procedure SaveToFile( AfileName: string)
22170: Procedure SAVETOFILE( const FILENAME : String)
22171: Procedure SaveToFile( const fileName : WideString)
22172: Procedure SaveToFile( const fileName : WideString; Format : TPersistFormat)
22173: Procedure SaveToFile( const fileName, FileType : string; Bitmap : TLinearBitmap)
22174: procedure SaveToFile(FileName: string);
22175: procedure SaveToFile(FileName:String)
22176: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
22177: Procedure SaveToStream(OutStream TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
22178: Procedure SaveToStream( S : TStream)
22179: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
22180: Procedure SaveToStream( Stream : TStream)
22181: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
22182: procedure SaveToStream(Stream: TStream);
22183: procedure SaveToStream(Stream:TStream)
22184: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
22185: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
22186: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
22187: procedure Say(const sText: string)
22188: Procedure SBytecode1Click( Sender : TObject)
22189: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
22190: procedure ScriptExplorer1Click(Sender: TObject);
22191: Procedure Scroll( Distance : Integer)
22192: Procedure Scroll( DX, DY : Integer)
22193: procedure ScrollBy(DeltaX, DeltaY: Integer);
22194: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
22195: Procedure ScrollTabs( Delta : Integer)
22196: Procedure Search1Click( Sender : TObject)
22197: procedure SearchAndOpenDoc(vfilenamepath: string)
22198: procedure SearchAndOpenFile(vfilenamepath: string)
22199: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
22200: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
22201: Procedure SearchNext1Click( Sender : TObject)
22202: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
22203: Procedure Select1( const Nodes : array of TTreeNode);
22204: Procedure Select2( Nodes : TList);
22205: Procedure SelectNext( Direction : Boolean)
22206: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
22207: Procedure SelfTestPEM //unit uPSI_cPEM
22208: Procedure Send( AMsg : TIdMessage)
22209: //config forst in const MAILINIFILE = 'maildef.ini';
22210: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
22211: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
22212: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
22213: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
22214: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
22215: Procedure SendResponse
22216: Procedure SendStream( AStream : TStream)
22217: procedure SendMCICmd(Cmd: string); !
22218: Procedure Set8087CW( NewCW : Word)
22219: Procedure SetAll( One, Two, Three, Four : Byte)
22220: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
22221: Procedure SetAppDispatcher( const AD dispatcher : TComponent)
22222: procedure SetArrayLength;
22223: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
22224: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
22225: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer); //all init
22226: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
22227: procedure SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: integer);'
22228: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);'
22229: procedure Set2dimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);'
22230: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
22231: procedure Set3dimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
22232: Procedure SetAsHandle( Format : Word; Value : THandle)
22233: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
22234: procedure SetCaptureControl(Control: TControl);
22235: Procedure SetColumnAttributes
22236: Procedure SetCookieField(Values:TStrings; const ADomain,APath:string;AExpires:TDateTime;ASecure:Bool)
22237: Procedure SetCustomHeader( const Name, Value : string)
22238: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts; const FieldName:Widestring)
22239: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
22240: Procedure SetFocus
22241: procedure SetFocus; virtual;
22242: Procedure SetInitialState
22243: Procedure SetKey

```

```

22244: procedure SetLastError(ErrorCode: Integer)
22245: procedure SetLength;
22246: procedure SetLength2(var S: string; NewLength: Integer)');
22247: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
22248: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
22249: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
22250: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
22251: Procedure SetParams1( UpdateKind : TUpdateKind);
22252: Procedure SetPassword( const Password : string)
22253: Procedure SetPointer( Ptr : Pointer; Size : Longint)
22254: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
22255: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
22256: Procedure SetProvider( Provider : TComponent)
22257: Procedure SetProxy( const Proxy : string)
22258: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
22259: Procedure SetRange( const StartValues, EndValues : array of const)
22260: Procedure SetRangeEnd
22261: Procedure SetRate( const aPercent, aYear : integer)
22262: procedure SetRate(const aPercent, aYear: integer)
22263: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
22264: Procedure SetSafeCallExceptionMsg( Msg : String)
22265: procedure SETSELTEXTBUF(BUFFER:PCHAR)
22266: Procedure SetSize( AWidth, AHeight : Integer)
22267: procedure SetSize(NewSize:LongInt)
22268: procedure SetString(var s: string; buffer: PChar; len: Integer)
22269: Procedure SetStrings( List : TStrings)
22270: Procedure SetText( Text : PwideChar)
22271: procedure SetText(Text: PChar);
22272: Procedure SetTextBuf( Buffer : PChar)
22273: procedure SETTEXTBUF(BUFFER:PCHAR)
22274: Procedure SetTick( Value : Integer)
22275: Procedure SetTimeout( ATimeOut : Integer)
22276: Procedure SetTraceEvent( Event : TDBXTraceEvent)
22277: Procedure SetUserName( const UserName : string)
22278: Procedure SetWallpaper( Path : string);
22279: procedure ShellStyle1Click(Sender: TObject);
22280: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
22281: Procedure ShowFileProperties( const FileName : string)
22282: Procedure ShowInclde1Click( Sender : TObject)
22283: Procedure ShowInterfaces1Click( Sender : TObject)
22284: Procedure ShowLastException1Click( Sender : TObject)
22285: Procedure ShowMessage( const Msg : string)
22286: Procedure ShowMessageBig(const aText : string);
22287: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
22288: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
22289: Procedure MsgBig(const aText : string); //alias
22290: procedure showmessage(mytext: string);
22291: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
22292: procedure ShowMessageFmt(const Msg: string; Params: array of const)
22293: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
22294: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
22295: Procedure ShowSearchDialog( const Directory : string)
22296: Procedure ShowSpecChars1Click( Sender : TObject)
22297: Procedure ShowBitmap(bitmap: TBitmap); //draw in a form!
22298: Procedure ShredFile( const FileName : string; Times : Integer)
22299: procedure Shuffle(v0: TStringList);
22300: Procedure ShuffleList( var List : array of Integer; Count : Integer)
22301: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
22302: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
22303: Procedure SinCose( X : Extended; out Sin, Cos : Extended)
22304: Procedure Site( const ACommand : string)
22305: Procedure SkipEOL
22306: Procedure Sleep( ATIME : cardinal)
22307: Procedure Sleep( milliseconds : Cardinal)
22308: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
22309: Procedure Slinenumbers1Click( Sender : TObject)
22310: Procedure Sort
22311: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
22312: procedure SoundAlarm; //beep seq
22313: procedure Speak(const sText: string) //async like voice
22314: procedure Speak2(const sText: string) //sync
22315: procedure Split(Str: string; SubStr: string; List: TStrings);
22316: Procedure SplitNameValuePair( const Line : string; var Name, Value : string)
22317: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
22318: Procedure SplitColumnsNoTrim(const AData: String; AStrings : TStrings; const ADelim : String)
22319: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
22320: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
22321: procedure SQLSyntax1Click(Sender: TObject);
22322: Procedure SRand( Seed : RNG_IntType)
22323: Procedure Start
22324: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
22325: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
//Ex. StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
22326: Procedure StartTransaction( TransDesc : TTransactionDesc)
22328: Procedure Status( var AStatusList : TStringList)
22329: Procedure StatusBar1DblClick( Sender : TObject)
22330: Procedure StepInto1Click( Sender : TObject)
22331: Procedure StepIt
22332: Procedure StepOut1Click( Sender : TObject)

```

```

22333: Procedure Stop
22334: procedure stopmp3;
22335: procedure StartWeb(aurl: string);
22336: Procedure Str(aint: integer; astr: string); //of system
22337: Procedure StrDispose( Str : PChar)
22338: procedure StrDispose(Str: PChar)
22339: Procedure StrReplace(var Str: String; Old, New: String);
22340: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
22341: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
22342: Procedure StringToBytes( Value : String; Bytes : array of byte)
22343: procedure StrSet(c : Char; I : Integer; var s : String);
22344: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
22345: Procedure StructureMount( APath : String)
22346: procedure STYLECHANGED(SENDER:TOBJECT)
22347: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
22348: procedure Succ(X: int64);
22349: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
22350: procedure SwapChar(var X,Y: char); //swapX follows
22351: Procedure SwapFloats( var X, Y : Float)
22352: procedure SwapGrid(grd: TStringGrid);
22353: Procedure SwapOrd( var I, J : Byte);
22354: Procedure SwapOrd( var X, Y : Integer)
22355: Procedure SwapOrd1( var I, J : Shortint);
22356: Procedure SwapOrd2( var I, J : Smallint);
22357: Procedure SwapOrd3( var I, J : Word);
22358: Procedure SwapOrd4( var I, J : Integer);
22359: Procedure SwapOrd5( var I, J : Cardinal);
22360: Procedure SwapOrd6( var I, J : Int64);
22361: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
22362: Procedure Synchronize( Method : TMethod);
22363: procedure SyntaxCheck1Click(Sender: TObject);
22364: Procedure SysFreeString(const S: WideString); stdcall;
22365: Procedure TakeOver( Other : TLinearBitmap)
22366: Procedure TalkIn(const sText: string) //async voice
22367: procedure tbtn6resClick(Sender: TObject);
22368: Procedure tbtnUseCaseClick( Sender : TObject)
22369: procedure TerminalStyle1Click(Sender: TObject);
22370: Procedure Terminate
22371: Procedure texSyntax1Click( Sender : TObject)
22372: procedure TextOut(X, Y: Integer; Text: string);
22373: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
22374: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
22375: Procedure TextRect( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
22376: Procedure TextStart
22377: procedure TILE
22378: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
22379: Procedure TitleClick( Column : TColumn)
22380: Procedure ToDo
22381: procedure Tone(500 hz, 10000 ms length);
22382: procedure toolbtnTutorialClick(Sender: TObject);
22383: Procedure Trace1( AURL : string; const AResponseContent : TStream);
22384: Procedure TransferMode( ATransferMode : TIIdFTPTransferMode)
22385: Procedure Truncate
22386: procedure Tutorial101Click(Sender: TObject);
22387: procedure Tutorial10Statistics1Click(Sender: TObject);
22388: procedure Tutorial11Forms1Click(Sender: TObject);
22389: procedure Tutorial12SQL1Click(Sender: TObject);
22390: Procedure tutorial11Click( Sender : TObject)
22391: Procedure tutorial21Click( Sender : TObject)
22392: Procedure tutorial31Click( Sender : TObject)
22393: Procedure tutorial4Click( Sender : TObject)
22394: Procedure Tutorial5Click( Sender : TObject)
22395: procedure Tutorial6Click(Sender: TObject);
22396: procedure Tutorial91Click(Sender: TObject);
22397: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
22398: procedure UniqueString(var str: AnsiString)
22399: procedure UnloadLoadPackage(Module: HMODULE)
22400: Procedure Unlock
22401: Procedure UNMERGE( MENU : TMAINMENU)
22402: Procedure UnRegisterChanges( Value : TChangeLink)
22403: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
22404: Procedure UnregisterConversionType( const AType : TConvType)
22405: Procedure UnRegisterProvider( Prov : TCustomProvider)
22406: Procedure UPDATE
22407: Procedure UpdateBatch( AffectRecords : TAffectRecords)
22408: Procedure UPDATECURSORPOS
22409: Procedure UpdateFile
22410: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
22411: Procedure UpdateResponse( AResponse : TWebResponse)
22412: Procedure UpdateScrollBar
22413: Procedure UpdateView1Click( Sender : TObject)
22414: procedure UpdateExeResource(Const Source,Dest:string); //!
22415: procedure Val(const s: string; var n, z: Integer)
22416: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
22417: Procedure VarFTMBcdCreate( var ADest : Variant; const ABCd : TBcd);
22418: Procedure VariantAdd( const src : Variant; var dst : Variant)
22419: Procedure VariantAnd( const src : Variant; var dst : Variant)
22420: Procedure VariantArrayRedim( var V : Variant; High : Integer)
22421: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)

```

```

22422: Procedure VariantClear( var V : Variant)
22423: Procedure VariantCpy( const src : Variant; var dst : Variant)
22424: Procedure VariantDiv( const src : Variant; var dst : Variant)
22425: Procedure VariantMod( const src : Variant; var dst : Variant)
22426: Procedure VariantMul( const src : Variant; var dst : Variant)
22427: Procedure VariantOr( const src : Variant; var dst : Variant)
22428: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
22429: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2 : integer);
22430: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3 : integer);
22431: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
22432: Procedure VariantPutElement5( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
22433: Procedure VariantShl( const src : Variant; var dst : Variant)
22434: Procedure VariantShr( const src : Variant; var dst : Variant)
22435: Procedure VariantSub( const src : Variant; var dst : Variant)
22436: Procedure VariantXor( const src : Variant; var dst : Variant)
22437: Procedure VarCastError;
22438: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
22439: Procedure VarInvalidOp
22440: Procedure VarInvalidNullOp
22441: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
22442: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
22443: Procedure VarArrayCreateError
22444: Procedure VarResultCheck( AResult : HRESULT);
22445: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
22446: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
22447: Function VarTypeAsText( const AType : TVarType) : string
22448: procedure Voice( const sText: string) //async
22449: procedure Voice2( const sText: string) //sync
22450: Procedure WaitMilliseconds( AMSec : word)
22451: Procedure WaitMS( AMSec : word)');
22452: procedure WebCampic(picname: string); //eg: c:\mypic.png
22453: Procedure WideAppend( var dst : WideString; const src : WideString)
22454: Procedure WideAssign( var dst : WideString; var src : WideString)
22455: Procedure WideDelete( var dst : WideString; index, count : Integer)
22456: Procedure WideFree( var s : WideString)
22457: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
22458: Procedure WideFromPChar( var dst : WideString; src : PChar)
22459: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
22460: Procedure WideStringLength( var dst : WideString; len : Integer)
22461: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
22462: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
22463: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
22464: Procedure Wininet_HttpGet( const Url: string; Stream:TStream);
22465: Procedure HttpGet( const Url: string; Stream:TStream);
22466: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
22467: Procedure WordWrap1Click( Sender : TObject)
22468: Procedure Write( const AOut : string)
22469: Procedure Write( Socket : TSocket)
22470: procedure Write(S: string);
22471: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
22472: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
22473: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
22474: procedure WriteBuffer(Buffer:String;Count:LongInt)
22475: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
22476: Procedure WriteChar( AValue : Char)
22477: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
22478: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
22479: Procedure WriteFloat( const Section, Name : string; Value : Double)
22480: Procedure WriteHeader( AHeader : TStrings)
22481: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
22482: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
22483: Procedure WriteLn( const AOut : string)
22484: procedure Writeln(s: string);
22485: Procedure WriteLog( const FileName, LogLine : string)
22486: Procedure WriteRFCReply( AReply : TIdRFCReply)
22487: Procedure WriteRFCStrings( AStrings : TStrings)
22488: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
22489: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
22490: Procedure WriteString( const Section, Ident, Value : String)
22491: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
22492: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
22493: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
22494: Procedure Write16bitResourceHeader(const AName: TBytes; DataSize : Integer; Output : TStream)
22495: Procedure Write32bitResourceHeader(const AName: TBytes; DataSize : Integer; Output : TStream)
22496: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
22497: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
22498: procedure XMLSyntax1Click(Sender: TObject);
22499: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
22500: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
22501: Procedure ZeroFillStream( Stream : TMemoryStream)
22502: procedure XMLSyntax1Click(Sender: TObject);
22503: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
22504: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
22505: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
22506: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
22507: procedure(Sender, Source: TObject; X, Y: Integer)
22508: procedure(Sender, Target: TObject; X, Y: Integer)
22509: procedure(Sender: TObject; ASection, AWidth: Integer)
22510: procedure(Sender: TObject; ScrollCode: TScrollCode; var ScrollPos: Integer)

```

```

22511: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
22512: procedure(Sender: TObject; var Action: TCloseAction)
22513: procedure(Sender: TObject; var CanClose: Boolean)
22514: procedure(Sender: TObject; var Key: Char);
22515: ProcedureName ProcedureNames ProcedureParametersCursor @
22516:
22517: *****Now Constructors constructor *****
22518: Size is: 1640 1588 1248 1115 996 628 550 544 501 459 (381)
22519: Attach(VersionInfoData : Pointer; Size : Integer)
22520: constructor Create( ABuckets : TBucketListSizes)
22521: Create( ACallBackWnd : HWnd)
22522: Create( AClient : TCustomTaskDialog)
22523: Create( AClient : TIdTelnet)
22524: Create( ACollection : TCollection)
22525: Create( ACollection : TFavoriteLinkItems)
22526: Create( ACollection : TTaskDialogButtons)
22527: Create( AConnection : TIdCustomHTTP)
22528: Create( ACreatSuspended : Boolean)
22529: Create( ADataSet : TCustomSQLDataSet)
22530: CREATE( ADATASET : TDATASET)
22531: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
22532: Create( AGrid : TCustomDBGrid)
22533: Create( AGrid : TStringGrid; AIndex : Longint)
22534: Create( AHTTP : TIdCustomHTTP)
22535: Create( AListItems : TListItems)
22536: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
22537: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
22538: Create( AOwner : TCommonCalendar)
22539: Create( AOwner : TComponent)
22540: CREATE( AOWNER : TCOMPONENT)
22541: Create( AOwner : TCustomListView)
22542: Create( AOwner : TCustomOutline)
22543: Create( AOwner : TCustomRichEdit)
22544: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
22545: Create( AOwner : TCustomTreeView)
22546: Create( AOwner : TIdUserManager)
22547: Create( AOwner : TListItems)
22548: Create(AOwner:TObj;Handle:hDBCICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
22549: CREATE( AOWNER : TPERSISTENT)
22550: Create( AOwner : TPersistent)
22551: Create( AOwner : TTable)
22552: Create( AOwner : TTreeNodes)
22553: Create( AOwner : TWinControl; const ClassName : string)
22554: Create( AParent : TIdCustomHTTP)
22555: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
22556: Create( AProvider : TBaseProvider)
22557: Create( AProvider : TCustomProvider);
22558: Create( AProvider : TDataSetProvider)
22559: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
22560: Create( ASocket : TSocket)
22561: Create( AStrings : TWideStrings)
22562: Create( AToolBar : TToolBar)
22563: Create( ATreeNodes : TTreeNodes)
22564: Create( Autofill : boolean)
22565: Create( AWebPageInfo : TAbstractWebPageInfo)
22566: Create( AWebRequest : TWebRequest)
22567: Create( Collection : TCollection)
22568: Create( Collection : TIdMessageParts; ABody : TStrings)
22569: Create( Collection : TIdMessageParts; const AFileName : TFileName)
22570: Create( Column : TColumn)
22571: Create( const AConvFamily : TConvFamily; const ADescription : string)
22572: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
22573: Create( const AConvFamily:TConvFamily;const ADescription:string;const AToCommonProc,
AFromComProc:TConversionProc)
22574: Create( const AInitialState : Boolean; const AManualReset : Boolean)
22575: Create( const ATabSet : TTabSet)
22576: Create( const Compensate : Boolean)
22577: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
22578: Create( const FileName : string)
22579: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize : Int64; const SecAttr : PSecurityAttributes);
22580: Create( const FileName : string; FileMode : WordfmShareDenyWrite)
22581: Create( const MaskValue : string)
22582: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
22583: Create( const Prefix : string)
22584: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
22585: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
22586: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
22587: Create( CoolBar : TCoolBar)
22588: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
22589: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
22590: Create( DataSet : TDataSet; const
Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : WideString;
DepFields : TBits; FieldMap : TFieldMap)
22591: Create( DBCtrlGrid : TDBCtrlGrid)
22592: Create( DSTableProducer : TDSTableProducer)
22593: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
22594: Create( ErrorCode : DBIResult)
22595: Create( Field : TBlobField; Mode : TBlobStreamMode)

```

```

22596: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
22597: Create( HeaderControl : TCustomHeaderControl)
22598: Create( HTTPRequest : TWebRequest)
22599: Create( iStart : integer; sText : string)
22600: Create( iValue : Integer)
22601: Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
22602: Create( MciErrNo : MCIERROr; const Msg : string)
22603: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
22604: Create( Message : string; ErrorCode : DBResult)
22605: Create( Msg : string)
22606: Create( NativeError, Context : string; ErrorCode, PrevError : Integer; E : Exception)
22607: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
22608: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
22609: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
22610: Create(oSorce:TniRegularExpression;xDestinat:TniRegularExpression;xCharacts:TCharSet;bLambda:bool)
22611: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
22612: Create( Owner : TCustomComboBoxEx)
22613: CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
22614: Create( Owner : TPersistent)
22615: Create( Params : TStrings) Create( Size : Cardinal)
22616: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
22617: Create( StatusBar : TCustomStatusBar)
22618: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
22619: Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
22620: Create(AHandle:Integer)
22621: Create(AOwner: TComponent); virtual;
22622: Create(const AURI : string)
22623: Create(FileName:String;Mode:Word)
22624: Create(Instance:THandle;ResName:String;ResType:PChar)
22625: Create(Stream : TStream)
22626: Create( ADataSet : TDataSet);
22627: Create( const FileHandle:THandle; const Name:string; Protect:Cardinal; const MaximumSize:Int64; const SecAttr:PSecurityAttributes);
22628: Create( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
22629: Create2( Other : TObject);
22630: CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
22631: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
22632: CreateFmt( MciErrNo : MCIERROr; const Msg : string; const Args : array of const)
22633: CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
22634: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
22635: CREATE(OWNER:TCOMPONENT; Dummy: Integer)
22636: CreateRes( Ident : Integer);
22637: CreateRes( MciErrNo : MCIERROr; Ident : Integer)
22638: CreateRes( ResStringRec : PResStringRec);
22639: CreateResHelp( Ident : Integer; AHelpContext : Integer);
22640: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
22641: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
22642: CreateSize( AWidth, AHeight : Integer)
22643: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
22644:
22645: -----
22646: unit UPSI_MathMax;
22647: -----
22648: CONSTS
22649: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
22650: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
22651: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
22652: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
22653: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
22654: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(PI)
22655: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
22656: PiJ: Float = 3.1415926535897932384626433832795; // PI
22657: PI: Extended = 3.1415926535897932384626433832795;
22658: PiOn2: Float = 1.570796326794896619231216916398; // PI / 2
22659: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
22660: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
22661: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
22662: Sqrt3: Float = 1.73205080756887729352747463415059; // Sqrt(3)
22663: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
22664: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
22665: Sqrtpi: Float = 1.7724538509055160272981674833411; // Sqrt(PI)
22666: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
22667: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
22668: ThreePi: Float = 9.4247779607693797153879301498385; // 3 * PI
22669: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
22670: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
22671: LnPi: Float = 1.1447298858494001741434273513531; // Ln(PI)
22672: Log2J: Float = 0.3010299956398119521373889472449; // Log10(2)
22673: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
22674: LogPi: Float = 0.4971498726941338543512682882909; // Log10(PI)
22675: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
22676: E: Float = 2.7182818284590452353602874713527; // Natural constant
22677: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
22678: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
22679: TwoToPower63: Float = 9223372036854775808.0; // 2^63
22680: GoldenMean: Float = 1.61803398874989488204586834365638; // GoldenMean
22681: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
22682: RadCor : Double = 57.29577951308232; {number of degrees in a radian}
22683: StDelta : Extended = 0.00001; {delta for difference equations}

```

```

22684: StEpsilon      : Extended = 0.00001; {epsilon for difference equations}
22685: StMaxIterations : Integer = 100;     {max attempts for convergence}
22686:
22687: procedure SIRegister_StdConvs(CL: TPSCompiler);
22688: begin
22689:   MetersPerInch = 0.0254; // [1]
22690:   MetersPerFoot = MetersPerInch * 12;
22691:   MetersPerYard = MetersPerFoot * 3;
22692:   MetersPerMile = MetersPerFoot * 5280;
22693:   MetersPerNauticalMiles = 1852;
22694:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
22695:   MetersPerLightSecond = 2.99792458E8; // [5]
22696:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
22697:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
22698:   MetersPerCubit = 0.4572; // [6][7]
22699:   MetersPerFathom = MetersPerFoot * 6;
22700:   MetersPerFurlong = MetersPerYard * 220;
22701:   MetersPerHand = MetersPerInch * 4;
22702:   MetersPerPace = MetersPerInch * 30;
22703:   MetersPerRod = MetersPerFoot * 16.5;
22704:   MetersPerChain = MetersPerRod * 4;
22705:   MetersPerLink = MetersPerChain / 100;
22706:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
22707:   MetersPerPica = MetersPerPoint * 12;
22708:
22709:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
22710:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
22711:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
22712:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
22713:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
22714:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
22715:
22716:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
22717:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
22718:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
22719:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
22720:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
22721:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
22722:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
22723:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
22724:
22725:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
22726:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
22727:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
22728:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
22729:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
22730:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
22731:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
22732:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
22733:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
22734:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
22735:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
22736:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
22737:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
22738:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
22739:
22740:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
22741:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
22742:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
22743:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
22744:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
22745:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
22746:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
22747:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
22748:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
22749:
22750:   GramsPerPound = 453.59237; // [1][7]
22751:   GramsPerDrams = GramsPerPound / 256;
22752:   GramsPerGrains = GramsPerPound / 7000;
22753:   GramsPerTons = GramsPerPound * 2000;
22754:   GramsPerLongTons = GramsPerPound * 2240;
22755:   GramsPerOunces = GramsPerPound / 16;
22756:   GramsPerStones = GramsPerPound * 14;
22757:
22758:   MaxAngle 9223372036854775808.0;
22759:   MaxTanH 5678.2617031470719747459655389854);
22760:   MaxFactorial( 1754);
22761:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
22762:   MinFloatingPoint(3.3621031431120935062626778173218E-4932);
22763:   MaxTanH( 354.89135644669199842162284618659);
22764:   MaxFactorial'LongInt'( 170);
22765:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
22766:   MinFloatingPointD(2.2250738585072013830902327173324E-308);
22767:   MaxTanH( 44.36141955583649980270285577323);
22768:   MaxFactorial'LongInt'( 33);
22769:   MaxFloatingPointS( 3.4028236692093846346337460743177E+38);
22770:   MinFloatingPointS( 1.1754943508222875079687365372222E-38);
22771:   PiExt( 3.1415926535897932384626433832795);
22772:   RatioDegToRad( PiExt / 180.0);

```

```

22773: RatioGradToRad( PiExt / 200.0);
22774: RatioDegToGrad( 200.0 / 180.0);
22775: RatioGradToDeg( 180.0 / 200.0);
22776: Crc16PolynomCCITT'LongWord $1021';
22777: Crc16PolynomIBM'LongWord $8005';
22778: Crc16Bits'LongInt'( 16);
22779: Crc16Bytes'LongInt'( 2);
22780: Crc16HighBit'LongWord $8000';
22781: NotCrc16HighBit','LongWord $7FFF';
22782: Crc32PolynomIEEE','LongWord $04C11DB7';
22783: Crc32PolynomCastagnoli','LongWord $1EDC6F41';
22784: Crc32Koopman','LongWord $741B8CD7';
22785: Crc32Bits','LongInt'( 32);
22786: Crc32Bytes','LongInt'( 4);
22787: Crc32HighBit','LongWord $80000000';
22788: NotCrc32HighBit','LongWord $7FFFFFFF';
22789:
22790: MinByte      = Low(Byte);
22791: MaxByte      = High(Byte);
22792: MinWord      = Low(Word);
22793: MaxWord      = High(Word);
22794: MinShortInt = Low(ShortInt);
22795: MaxShortInt = High(ShortInt);
22796: MinSmallInt = Low(SmallInt);
22797: MaxSmallInt = High(SmallInt);
22798: MinLongWord  = LongWord(Low(LongWord));
22799: MaxLongWord  = LongWord(High(LongWord));
22800: MinLongInt   = LongInt(Low(LongInt));
22801: MaxLongInt   = LongInt(High(LongInt));
22802: MinInt64    = Int64(Low(Int64));
22803: MaxInt64    = Int64(High(Int64));
22804: MinInteger  = Integer(Low(Integer));
22805: MaxInteger  = Integer(High(Integer));
22806: MinCardinal = Cardinal(Low(Cardinal));
22807: MaxCardinal = Cardinal(High(Cardinal));
22808: MinNativeUInt= NativeUInt(Low(NativeUInt));
22809: MaxNativeUInt= NativeUInt(High(NativeUInt));
22810: MinNativeInt = NativeInt(Low(NativeInt));
22811: MaxNativeInt = NativeInt(High(NativeInt));
22812: Function CosH( const Z : Float) : Float;
22813: Function SinH( const Z : Float) : Float;
22814: Function TanH( const Z : Float) : Float;
22815:
22816: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
22817: InvLn2      = 1.44269504088896340736; { 1/Ln(2) }
22818: InvLn10     = 0.43429448190325182765; { 1/Ln(10) }
22819: TwoPi       = 6.28318530717958647693; { 2*Pi }
22820: PiDiv2     = 1.5707963267949661923; { Pi/2 }
22821: SqrtPi     = 1.77245385090551602730; { Sqrt(Pi) }
22822: Sqrt2Pi    = 2.50662827463100050242; { Sqrt(2*Pi) }
22823: InvSqrt2Pi = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
22824: LnSqrt2Pi  = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
22825: Ln2PiDiv2  = 0.91893853320467274178; { Ln(2*Pi)/2 }
22826: Sqrt2      = 1.41421356237309504880; { Sqrt(2) }
22827: Sqrt2Div2  = 0.70710678118654752440; { Sqrt(2)/2 }
22828: Gold       = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
22829: CGold      = 0.38196601125010515179; { 2 - GOLD }
22830: MachEp     = 2.220446049250313E-16; { 2^{(-52)} }
22831: MaxNum     = 1.797693134862315E+308; { 2^{1024} }
22832: MinNum     = 2.225073858507202E-308; { 2^{(-1022)} }
22833: MaxLog     = 709.7827128933840;
22834: MinLog     = -708.3964185322641;
22835: MaxFac     = 170;
22836: MaxGam     = 171.624376956302;
22837: MaxLgm     = 2.556348E+305;
22838: SingleCompareDelta = 1.0E-34;
22839: DoubleCompareDelta = 1.0E-280;
22840: {#IFDEF CLR}
22841: ExtendedCompareDelta = DoubleCompareDelta;
22842: {#ELSE}
22843: ExtendedCompareDelta = 1.0E-4400;
22844: {#ENDIF}
22845: Bytes1KB   = 1024;
22846: Bytes1MB   = 1024 * Bytes1KB;
22847: Bytes1GB   = 1024 * Bytes1MB;
22848: Bytes64KB  = 64 * Bytes1KB;
22849: Bytes64MB  = 64 * Bytes1MB;
22850: Bytes2GB   = 2 * LongWord(Bytes1GB);
22851: clBlack32', $FF000000 );
22852: clDimGray32', $FF3F3F3F );
22853: clGray32', $FF7F7F7F );
22854: clLightGray32', $FFBFBFBF );
22855: clWhite32', $FFFFFF );
22856: clMaroon32', $FF7F0000 );
22857: clGreen32', $FF007F00 );
22858: clOlive32', $FF7F7F00 );
22859: clNavy32', $FF00007F );
22860: clPurple32', $FF7F007F );
22861: clTeal32', $FF007F7F );

```

```
22862:    clRed32', $FFFF0000 ));
22863:    clLime32', $FF00FF00 ));
22864:    clYellow32', $FFFFFF00 ));
22865:    clBlue32', $FF0000FF ));
22866:    clFuchsia32', $FFFF00FF ));
22867:    clAqua32', $FF00FFFF ));
22868:    clAliceBlue32', $FFF0F8FF ));
22869:    clAntiqueWhite32', $FFFAEBD7 ));
22870:    clAquamarine32', $FF7FFF04 ));
22871:    clAzure32', $FFF0FFFF ));
22872:    clBeige32', $FFF5F5DC ));
22873:    clBisque32', $FFF8E4C4 ));
22874:    clBlancheDalmond32', $FFFEBECD ));
```

22875: clBlueViolet32', \$FF8A2BE2 ));  
22876: clBrown32', \$FFA52A2A ));  
22877: clBurlyWood32', \$FFD8B887 ));  
22878: clCadetblue32', \$FF5F9EA0 ));  
22879: clChartreuse32', \$FF7FFF00 ));  
22880: clChocolate32', \$FFD2691E ));  
22881: clCoral32', \$FFFF7F50 ));  
22882: clCornFlowerBlue32', \$FF6495ED ));  
22883: clCornSilk32', \$FFFFF8DC ));  
22884: clCrimson32', \$FFDC143C ));  
22885: clDarkBlue32', \$FF00008B ));  
22886: clDarkCyan32', \$FF008B8B ));  
22887: clDarkGoldenRod32', \$FFB8860B ));  
22888: clDarkGray32', \$FFA9A9A9 ));  
22889: clDarkGreen32', \$FF006400 ));  
22890: clDarkGrey32', \$FFA9A9A9 ));  
22891: clDarkKhaki32', \$FFBDB76B ));  
22892: clDarkMagenta32', \$FF8B008B ));  
22893: clDarkOliveGreen32', \$FF556B2F ));  
22894: clDarkOrange32', \$FFF8C00 ));  
22895: clDarkOrchid32', \$FF9932CC ));  
22896: clDarkRed32', \$FF8B0000 ));  
22897: clDarkSalmon32', \$FFE9967A ));  
22898: clDarkSeaGreen32', \$FF8FB8CF ));  
22899: clDarkSlateBlue32', \$FF483D8B ));  
22900: clDarkSlateGray32', \$FF2F4F4F ));  
22901: clDarkSlateGrey32', \$FF2F4F4F ));  
22902: clDarkTurquoise32', \$FF00CED1 ));  
22903: clDarkViolet32', \$FF9400D3 ));  
22904: clDeepPink32', \$FFFF1493 ));  
22905: clDeepSkyBlue32', \$FF00BFFF ));  
22906: clDodgerBlue32', \$FF1E90FF ));  
22907: clFireBrick32', \$FFB22222 ));  
22908: clFloralWhite32', \$FFFFFA00 ));  
22909: clGainsboro32', \$FFCDCDCD ));  
22910: clGhostWhite32', \$FFF8F8FF ));  
22911: clGold32', \$FFFFD700 ));  
22912: clGoldenRod32', \$FFDA520 ));  
22913: clGreenYellow32', \$FFADFF2F ));  
22914: clGrey32', \$FF808080 ));  
22915: clHoneyDew32', \$FFF0FF00 ));  
22916: clHotPink32', \$FFFF69B4 ));  
22917: clIndianRed32', \$FFCD5C5C ));  
22918: clIndigo32', \$FF4B0082 ));  
22919: clIvory32', \$FFFFFF00 ));  
22920: clKhaki32', \$FFFOE68C ));  
22921: clLavender32', \$FFE6E6FA ));  
22922: clLavenderBlush32', \$FFFF0F5 ));  
22923: clLawnGreen32', \$FF7CEC00 ));  
22924: clLemonChiffon32', \$FFFFFACD ));  
22925: clLightBlue32', \$FFADD8E6 ));  
22926: clLightCoral32', \$FFFO8080 ));  
22927: clLightCyan32', \$FFEOFFFF ));  
22928: clLightGoldenRodYellow32', \$FFFAFAD2 ));  
22929: clLightGreen32', \$FF90EE90 ));  
22930: clLightGrey32', \$FFD3D3D3 ));  
22931: clLightPink32', \$FFFFB6C1 ));  
22932: clLightSalmon32', \$FFFA07A ));  
22933: clLightSeagreen32', \$FF20B2AA ));  
22934: clLightSkyblue32', \$FF87CEFA ));  
22935: clLightSlategray32', \$FF778899 ));  
22936: clLightSlategrey32', \$FF778899 ));  
22937: clLightSteelblue32', \$FFB0C4DE ));  
22938: clLightYellow32', \$FFFFFFE0 ));  
22939: clLtGray32', \$FFC0C0C0 ));  
22940: clMedGray32', \$FFA0AOA4 ));  
22941: clDkGray32', \$FF808080 ));  
22942: clMoneyGreen32', \$FFC0DCC0 ));  
22943: clLegacySkyBlue32', \$FFA6CAF0 ));  
22944: clCream32', \$FFFFFBF0 ));  
22945: clLimeGreen32', \$FF32CD32 ));  
22946: clLinen32', \$FFFAF0E6 ));  
22947: clMediumAquamarine32', \$FF66CDAA ));  
22948: clMediumBlue32', \$FF0000CD ));  
22949: clMediumOrchid32', \$FFBA55D3 ));  
22950: clMediumPurple32', \$FF9370DB ));

```

22951:     clMediumSeaGreen32', $FF3CB371 ));
22952:     clMediumSlateBlue32', $FF7B68EE ));
22953:     clMediumSpringGreen32', $FF00FA9A ));
22954:     clMediumTurquoise32', $FF48D1CC ));
22955:     clMediumVioletRed32', $FFC71585 ));
22956:     clMidnightBlue32', $FF191970 ));
22957:     clMintCream32', $FFF5FFFA ));
22958:     clMistyRose32', $FFFFE4E1 ));
22959:     clMoccasin32', $FFFFE4B5 ));
22960:     clNavajoWhite32', $FFFFDEAD ));
22961:     clOldLace32', $FFFDF5E6 ));
22962:     clOliveDrab32', $FF6B8E23 ));
22963:     clOrange32', $FFFFA500 ));
22964:     clOrangeRed32', $FFFF4500 ));
22965:     clOrchid32', $FFDA70D6 ));
22966:     clPaleGoldenRod32', $FFEEE8AA ));
22967:     clPaleGreen32', $FF98FB98 ));
22968:     clPaleTurquoise32', $FFAFEEEE ));
22969:     clPaleVioletred32', $FFDB7093 ));
22970:     clPapayaWhip32', $FFFFEF05 ));
22971:     clPeachPuff32', $FFFFDAB9 ));
22972:     clPeru32', $FFCD853F ));
22973:     clPlum32', $FFDDA0DD ));
22974:     clPowderBlue32', $FFB0E0E6 ));
22975:     clRosyBrown32', $FFBC8F8F ));
22976:     clRoyalBlue32', $FF4169E1 ));
22977:     clSaddleBrown32', $FF8B4513 ));
22978:     clSalmon32', $FFFA8072 ));
22979:     clSandyBrown32', $FFF4A460 ));
22980:     clSeaGreen32', $FF2E8B57 ));
22981:     clSeashell32', $FFFFFF5EE ));
22982:     clSienna32', $FFA0522D ));
22983:     clSilver32', $FFC0C0C0 ));
22984:     clSkyblue32', $FF87CEEB ));
```

22985: clSlateBlue32', \$FF6A5ACD ));  
 22986: clSlateGray32', \$FF708090 ));  
 22987: clSlateGrey32', \$FF708090 ));  
 22988: clSnow32', \$FFFFFFFAFA ));  
 22989: clSpringgreen32', \$FF00FF7F ));  
 22990: clSteelblue32', \$FF4682B4 ));  
 22991: clTan32', \$FFD2B48C ));  
 22992: clThistle32', \$FFD8BF08 ));  
 22993: clTomato32', \$FFFF6347 ));  
 22994: clTurquoise32', \$FF40E0D0 ));  
 22995: clViolet32', \$FFEE82EE ));  
 22996: clWheat32', \$FFF5DEB3 ));  
 22997: clWhitesmoke32', \$FFF5F5F5 ));  
 22998: clYellowgreen32', \$FF9ACD32 ));  
 22999: clTrWhite32', \$7FFFFFFF ));  
 23000: clTrBlack32', \$7F000000 ));  
 23001: clTrRed32', \$7FFF0000 ));  
 23002: clTrGreen32', \$7F00FF00 ));  
 23003: clTrBlue32', \$7F0000FF ));  
 23004: // Fixed point math constants  
 23005: FixedOne = \$10000; FixedHalf = \$7FFF;  
 23006: FixedPI = Round(PI \* FixedOne);  
 23007: FixedToFloat = 1/FixedOne;  
 23008:  
 23009: Special Types  
 23010: \*\*\*\*  
 23011: type Complex = record //for complex numbers  
 23012: X, Y : Float;  
 23013: end;  
 23014: type TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R : '  
 23015: + Float; Theta : Float; end');  
 23016: type TVector = array of Float;  
 23017: TIntVector = array of Integer;  
 23018: TCompVector = array of Complex;  
 23019: TBoolVector = array of Boolean;  
 23020: TStringVector = array of String;  
 23021: TMatrix = array of TVector;  
 23022: TIntMatrix = array of TIntVector;  
 23023: TCompMatrix = array of TCompVector;  
 23024: TBoolMatrix = array of TBoolVector;  
 23025: TStringMatrix = array of TString;  
 23026: TByteArray = array[0..32767] of byte; !  
 23027: THexArray = array [0..15] of Char; // = '0123456789ABCDEF';  
 23028: TBitmapStyle = (bsNormal, bsCentered, bsStretched);  
 23029: T2StringArray = array of array of string;  
 23030: T2IntegerArray = array of array of integer;  
 23031: AddTypes('INT\_PTR', 'Integer  
 23032: AddTypes('LONG\_PTR', 'Integer  
 23033: AddTypes('UINT\_PTR', 'Cardinal  
 23034: AddTypes('ULONG\_PTR', 'Cardinal  
 23035: AddTypes('DWORD\_PTR', 'ULONG\_PTR  
 23036: TIntegerDynArray', 'array of Integer  
 23037: TCardinalDynArray', 'array of Cardinal  
 23038: TWordDynArray', 'array of Word  
 23039: TSmallIntDynArray', 'array of SmallInt

```

23040: TByteDynArray', 'array of Byte
23041: TShortIntDynArray', 'array of ShortInt
23042: TInt64DynArray', 'array of Int64
23043: TLongWordDynArray', 'array of LongWord
23044: TSingleDynArray', 'array of Single
23045: TDoubleDynArray', 'array of Double
23046: TBooleanDynArray', 'array of Boolean
23047: TStringDynArray', 'array of string
23048: TWideStringDynArray', 'array of WideString
23049: TDynByteArray = array of Byte;
23050: TDynShortintArray = array of Shortint;
23051: TDynSmallintArray = array of Smallint;
23052: TDynWordArray = array of Word;
23053: TDynIntegerArray = array of Integer;
23054: TDynLongintArray = array of Longint;
23055: TDynCardinalArray = array of Cardinal;
23056: TDynInt64Array = array of Int64;
23057: TDynExtendedArray = array of Extended;
23058: TDynDoubleArray = array of Double;
23059: TDynSingleArray = array of Single;
23060: TDynFloatArray = array of Float;
23061: TDynPointerArray = array of Pointer;
23062: TDynStringArray = array of string;
23063: TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
23064:           ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
23065: TSynSearchOptions = set of TSynSearchOption;
23066: TFloat = single
23067: Float = double
23068:
23069:
23070: /* Project: IFSI_WinForm1puzzle.pas BaseInclude RunTimeLib for maxbox *: pas_includebox.inc
23071: -----
23072: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
23073: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
23074: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
23075: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
23076: function CheckStringSum(vString: string): integer;
23077: function HexToInt(HexNum: string): LongInt;
23078: function IntToBin(Int: Integer): String;
23079: function BinToInt(Binary: String): Integer;
23080: function HexToBin(HexNum: string): string; external2
23081: function BinToHex(Binary: String): string;
23082: function IntToFloat(i: Integer): double;
23083: function AddThousandSeparator(S: string; myChr: Char): string;
23084: function Max3(const X,Y,Z: Integer): Integer;
23085: procedure Swap(var X,Y: char); // faster without inline
23086: procedure ReverseString(var S: String);
23087: function CharToHexStr(Value: Char): string;
23088: function CharToUniCode(Value: Char): string;
23089: function Hex2Dec(Value: Str002): Byte;
23090: function HexStrCodeToStr(Value: string): string;
23091: function HexToStr(i: integer; value: string): string;
23092: function UniCodeToStr(Value: string): string;
23093: function CRC16(statement: string): string;
23094: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
23095: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
23096: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
23097: Procedure ExecuteCommand(executeFile, paramstring: string);
23098: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
23099: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
23100: procedure SearchAndOpenDoc(vfilenamepath: string);
23101: procedure ShowInterfaces(myFile: string);
23102: function Fact2(av: integer): extended;
23103: Function BoolToStr(B: Boolean): string;
23104: Function GCD(x, y : LongInt) : LongInt;
23105: function LCM(m,n: longint): longint;
23106: function GetASCII: string;
23107: function GetItemHeight(Font: TFont): Integer;
23108: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
23109: function myGetWindowsDirectory(lpBufer: PChar; uSize: longword): longword;
23110: function getHINSTANCE: longword;
23111: function getHMODULE: longword;
23112: function GetASCII: string;
23113: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
23114: function WordIsOk(const AWord: string; var VW: Word): boolean;
23115: function TwentyFourBitValueisOk(const AValue: string; var VI: Integer): boolean;
23116: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
23117: function SafeStr(const s: string): string;
23118: function ExtractUrlPath(const FileName: string): string;
23119: function ExtractUrlName(const FileName: string): string;
23120: function IsInternet: boolean;
23121: function RotateLeft1Bit_u32( Value: uint32): uint32;
23122: procedure LinearRegression(const KnownY:array of Double;const KnownX:array of Double;NData:Int;var
23123:   LF:TStLinEst; ErrorStats: Bool);
23124: procedure getEnvironmentInfo;
23125: procedure AntiFreeze;
23126: function GetCPUSpeed: Double;
23127: function IsVirtualPcGuest : Boolean;
23128: function IsVmWareGuest : Boolean;

```

```

23128: procedure StartSerialDialog;
23129: function IsWoW64: boolean;
23130: function IsWow64String(var s: string): Boolean;
23131: procedure StartThreadDemo;
23132: Function RGB(R,G,B: Byte): TColor;
23133: Function Sendln(amess: string): boolean;
23134: Procedure maXbox;
23135: Function AspectRatio(aWidth, aHeight: Integer): String;
23136: function wget(aURL, afile: string): boolean;
23137: procedure PrintList(Value: TStringList);
23138: procedure PrintImage aValue: TBitmap; Style: TBitmapStyle);
23139: procedure getEnvironmentInfo;
23140: procedure AntiFreeze;
23141: function getBitmap(apath: string): TBitmap;
23142: procedure ShowMessageBig(const aText : string);
23143: function YesNoDialog(const ACaption, AMsg: string): boolean;
23144: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
23145: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
23146: //function myStrToBytes(const Value: String): TBytes;
23147: //function myBytesToStr(const Value: TBytes): String;
23148: function SaveAsExcelFile(AGrid:TStringGrid;ASheetName,AFileName:string;open:boolean):Bool;
23149: function getBitmap(apath: string): TBitmap;
23150: procedure ShowMessageBig(const aText : string);
23151: Function StrToBytes(const Value: String): TBytes;
23152: Function BytesToStr(const Value: TBytes): String;
23153: function SaveAsExcelFile(AGrid: TStringGrid;ASheetName,AFileName: string;open:boolean):Bool;
23154: function ReverseDNSLookup (const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var HostName:String):Bool;
23155: function FindInPaths(const fileName, paths : String) : String;
23156: procedure initHexArray(var hexn: THexArray);
23157: function josephusG(n,k: integer; var graphout: string): integer;
23158: function isPowerof2(num: int64): boolean;
23159: function powerOf2(exponent: integer): int64;
23160: function getBigPI: string;
23161: procedure MakeSound(Frequency[Hz],Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:string);
23162: function GetASCIILine: string;
23163: procedure MakeComplexSound(N:integer{stream # to use};freqlist:TStrings;Duration{mSec}: Integer;
23164:                                pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
23165: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
23166: procedure AddComplexSoundObjectToList(newf,newp,newa,neww,integer; freqlist: TStrings);
23167: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
23168: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
23169: function isKeyPressed: boolean;
23170: function Keypress: boolean;
23171: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
23172: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
23173: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
23174: function GetOSName: string;
23175: function GetOSVersion: string;
23176: function GetOSNumber: string;
23177: function getEnvironmentString: string;
23178: procedure StrReplace(var Str: String; Old, New: String);
23179: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
23180: function getTeamViewerID: string;
23181: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
23182: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
23183: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
23184: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
23185: function StartSocketService: Boolean;
23186: procedure StartSocketServiceForm;
23187: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
23188: function GetFileList1(apath: string): TStringlist;
23189: procedure LetFileList(FileList: TStringlist; apath: string);
23190: procedure StartWeb(aurl: string);
23191: function GetTodayFiles(startdir, amask: string): TStringlist;
23192: function PortTCPISOpen(dwPort : Word; ipAddressStr: String): boolean;
23193: function JavahashCode(val: string): Integer;
23194: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
23195: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
23196: Procedure HideWindowForSeconds(secs: integer); //3 seconds
23197: Procedure HideWindowForSeconds2(secs: integer; appHandle, aSelf: TForm); //3 seconds
23198: Procedure ConvertToGray(Cnv: TCanvas);
23199: function GetFileDate(aFile:string; aWithTime:Boolean):string;
23200: procedure ShowMemory;
23201: function ShowMemory2: string;
23202: function getHostIP: string;
23203: procedure ShowBitmap(bmap: TBitmap);
23204: function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
23205: function CreateDBGridForm(dblist: TStringList): TListbox;
23206: function isService: boolean;
23207: function isApplication: boolean;
23208: function isTerminalSession: boolean;
23209: function SetPrivilege(privilegeName: string; enable: boolean): boolean;
23210: procedure getScriptandRunAsk;
23211: procedure getScriptandRun(ascript: string);
23212: function VersionCheckAct: string;
23213: procedure getBox(aurl, extension: string);
23214: function CheckBox: string;
23215: function isNTFS: boolean;

```

```

23216: //procedure doWebCamPic;
23217: procedure doWebCamPic(picname: string);
23218: function readm: string;
23219: procedure GetGEOMapandRunAsk; //APIKey needed
23220: function GetMapX(C_form,apath: string; const Data: string): boolean;
23221: procedure GetGEOMap(C_form,apath: string; const Data: string);
23222: function GetMapXGeoReverse(C_form: string; const lat,long: string): string;
23223: //function RoundTo(const AValue: Extended;
23224: //      const ADigit: TRoundToEXRangeExtended): Extended;
23225: function GetGeocodeCoord(C_form: string; const data:string; atxt: boolean): string;
23226: function GetGeoCoord(C_form: string; const data:string; atxt: boolean): string;
23227: ex.: writeln(GetGeoCoord('xml','church cefalu sicily',true))
23228: function DownloadFile(SourceFile, DestFile: string): Boolean;
23229: function DownloadfileOpen(Sourcefile, DestFile: string): Boolean;
23230: function OpenMap(const Data: string): boolean;
23231: function GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
23232: Function getFileCount(amask: string): integer;
23233: function CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TNavPos): string;
23234: procedure Debugln(DebugLOGFILE: string; E: string);
23235: function IntToFloat(i: Integer): double;
23236: function AddThousandsSeparator(S: string; myChr: Char): string;
23237: function mymcisSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
23238: function RoundTime(ADate: string; Rounding: Integer; bRound: Boolean): string;
23239: function DynamicDllCallName(Dll: String; const Name: String; HasResult: Boolean; var Returned: Cardinal;
23240: const Parameters: array of integer): Boolean;'{';
23241: function DynamicDllCall(Dll: String; const Name: String; HasResult: Boolean; var Returned: Cardinal;
23242: const Parameters: array of integer): Boolean;'{';
23243: 
23244: End C:\maXbook\maxbox3\mX3999\maxbox3\source\IFSI_WinForm1puzzle.pas File loaded
23245: 
23246: // News of 3.9.8 up
23247: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
23248: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
23249: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
23250: JvChart - TJvChart Component - 2009 Public
23251: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
23252: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
23253: TAdoQuery.SQL.Add() fixed, ShlwAPI extensions, Indy HTTPHeader Extensions
23254: DMath DLL included incl. Demos
23255: Interface Navigator menu/View/Intf Navigator
23256: Unit Explorer menu/Debug/Units Explorer
23257: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
23258: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
23259: Script History to 9 Files WebServer light /Options/Addons/WebServer
23260: Full Text Finder, JVSimLogic Simulator Package
23261: Halt-Stop Program in Menu, WebServer2, Stop Event ,
23262: Conversion Routines, Prebuild Forms, CodeSearch
23263: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
23264: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
23265: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
23266: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
23267: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
23268: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
23269: IDE Reflection API, Session Service Shell S3
23270: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
23271: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
23272: arduino map() function, PRandom Generator
23273: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
23274: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
23275: REST Test Lib, Multilang Component, Forth Interpreter
23276: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
23277: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
23278: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
23279: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
23280: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
23281: QRCode Service, add more CFunctions like CDatetime of Synapse
23282: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
23283: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
23284: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
23285: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
23286: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
23287: BOLD Package, Indy Package5, matrix. MATHEMAX
23288: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
23289: emax layers: system-package-component-unit-class-function-block
23290: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
23291: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
23292: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
23293: OpenGL Game Demo: ..Options/Add Ons/Reversi
23294: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
23295: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
23296: 7% performance gain (hot spot profiling)
23297: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
23298: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
23299: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
23300: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
23301: 
23302: add routines in 3.9.7.5

```

```

23303: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSEexec);
23304: 996: procedure RIRegister_DBCtrls_Routines(S: TPSEexec);
23305: 069: procedure RIRegister_IdStrings_Routines(S: TPSEexec);
23306: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSEexec);
23307: 215: procedure RIRegister_PNGLoader_Routines(S: TPSEexec);
23308: 374: procedure RIRegister_SerDlgls_Routines(S: TPSEexec);
23309: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSEexec);
23310: 1216 procedure RIRegistger_uPSI_KDialogs, TKBrowseFolderDialog;
23311:
23312: ////////////////////////////////////////////////////////////////// TestUnits //////////////////////////////////////////////////////////////////
23313: SelftestPEM;
23314: SelfTestCFundamentUtils;
23315: SelfTestCFileUtils;
23316: SelfTestCDatetime;
23317: SelfTestCTimer;
23318: SelfTestCRandom;
23319: SelftestAES;
23320: SelfTestASN1;
23321: SelfTestX509;
23322: TestDes: boolean;
23323: Test3Des: boolean;
23324: TestAes: boolean;
23325: SelfTestcTLSUtils;
23326: SelfTestCFundamentUtils;
23327: SelfTestcHTTPUtils
23328: SelfTestcXMLFunctions
23329:
23330:
23331: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter'
23332:           Assert(WinPathToUnixPath('\c\d\f') = '/c/d/f', 'WinPathToUnixPath'
23333:
23334: // Note: There's no need for installing a client certificate in the
23335: //        webbrowser. The server asks the webbrowser to send a certificate but
23336: //        if nothing is installed the software will work because the server
23337: //        doesn't check to see if a client certificate was supplied. If you want you can install: file:
23338: c_cacert.p12 password: c_cakey
23339: TKObject = class(TObject)
23340: private
23341:   FParent: TKObjectList;
23342:   procedure SetParent(const Value: TKObjectList);
23343: protected
23344:   FUpdateLock: Integer;
23345:   procedure CallBeforeUpdate; virtual;
23346:   procedure CallAfterUpdate; virtual;
23347:   procedure ParentChanged; virtual;
23348: public
23349:   constructor Create; virtual;
23350:   procedure Assign(ASource: TKObject); virtual;
23351:   function EqualProperties(AValue: TKObject): Boolean; virtual;
23352:   procedure LockUpdate; virtual;
23353:   procedure UnLockUpdate; virtual;
23354:   function UpdateUnlocked: Boolean; virtual;
23355:   property Parent: TKObjectList read FParent write SetParent;
23356: end;
23357:
23358: TKObjectClass = class of TKObject;
23359: TKObjectList = class(TObjectList)
23360: protected
23361:   FUpdateLock: Integer;
23362:   procedure CallBeforeUpdate; virtual;
23363:   procedure CallAfterUpdate; virtual;
23364: public
23365:   constructor Create; virtual;
23366:   function Add(AObject: TObject): Integer;
23367:   procedure Assign(ASource: TKObjectList); virtual;
23368:   function EqualProperties(AValue: TKObjectList): Boolean; virtual;
23369:   procedure Insert(Index: Integer; AObject: TObject);
23370:   procedure LockUpdate; virtual;
23371:   procedure UnLockUpdate; virtual;
23372:   function UpdateUnlocked: Boolean; virtual;
23373: end;
23374:
23375: TGraphicControl = class(TControl)
23376: private
23377:   FCanvas: TCanvas;
23378:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
23379: protected
23380:   procedure Paint; virtual;
23381:   property Canvas: TCanvas read FCanvas;
23382: public
23383:   constructor Create(AOwner: TComponent); override;
23384:   destructor Destroy; override;
23385: end;
23386:
23387: TCustomControl = class(TWinControl)
23388: private
23389:   FCanvas: TCanvas;
23390:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;

```

```

23391: protected
23392:   procedure Paint; virtual;
23393:   procedure PaintWindow(DC: HDC); override;
23394:   property Canvas: TCanvas read FCanvas;
23395: public
23396:   constructor Create(AOwner: TComponent); override;
23397:   destructor Destroy; override;
23398: end;
23399: RegisterPublishedProperties;
23400: ('ONCHANGE', 'TNotifyEvent', iptrw);
23401: ('ONCLICK', 'TNotifyEvent', iptrw);
23402: ('ONDBLCLICK', 'TNotifyEvent', iptrw);
23403: ('ONENTER', 'TNotifyEvent', iptrw);
23404: ('ONEXIT', 'TNotifyEvent', iptrw);
23405: ('ONKEYDOWN', 'TKeyEvent', iptrw);
23406: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
23407: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
23408: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
23409: ('ONMOUSEUP', 'TMouseEvent', iptrw);
23410: //*****
23411: // To stop the while loop, click on Options/Show Include (boolean switch) !
23412: Control a loop in a script with a form event:
23413: IncludeON; //control the while loop
23414: while maxform1.ShowInclude1.checked do begin //menu event Options/Show Include
23415: repeat {for it:= 1 to n do} until is keypressed //keypress in output window below (memo2)
23416:
23417: //-----
23418: //*****mX4 ini-file Configuration*****
23419: //-----
23420: using config file maxboxdef.ini      menu/Help/Config File
23421:
23422: //*** Definitions for maXbox mx3 ***
23423: [FORM]
23424: LAST_FILE=E:\maxbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
23425: FONTSIZE=14
23426: EXTENSION=txt
23427: SCREENX=1386
23428: SCREENY=1077
23429: MEMHEIGHT=350
23430: PRINTFONT=Courier New //GUI Settings
23431: LINENUMBERS=Y //line numbers at gutter in editor at left side
23432: EXCEPTIONLOG=Y //store excepts + success in 2 log files see below! -menu Debug>Show Last Exceptions
23433: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
23434: BOOTSCRIPT=Y //enabling load a boot script
23435: MEMORYREPORT=Y //shows memory report on closing maXbox
23436: MACRO=Y //expand macros (see below) incode e.g. #path:E:\maxbox3\mXGit3998\maxbox3\docs\
23437: NAVIGATOR=N //shows function list at the right side of editor
23438: NAVWIDTH=350 //width of the right side interface list <CTRL L> >=200
23439: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
23440: [WEB]
23441: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
23442: IPHOST=192.168.1.53 //run as Administrator!
23443: ROOTCERT=filepathY
23444: SCERT=filepathY
23445: RSAKEY=filepathY
23446: VERSIONCHECK=Y
23447: APP=C:\WINDOWS\System32\calc.exe //set path to an external app
23448: MYSCRIPT=E:\maxbox3\mXGit3999\maxbox3\examples\330_myclock.txt //start script of menu /View/MyScript
23449:
23450: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
23451: Also possible to set report memory in script to override ini setting
23452: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
23453:
23454: After Change the ini file you can reload the file with ..//Help/Config Update
23455: //-----
23456: //*****mX4 maildef.ini ini-file Configuration*****
23457: //-----
23458: //*** Definitions for maXMail ***
23459: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
23460: [MAXMAIL]
23461: HOST=getmail.softwareschule.ch
23462: USER=mailusername
23463: PASS=password
23464: PORT=110
23465: SSL=Y
23466: BODY=Y
23467: LAST=5
23468:
23469: ADO Connection String:
23470: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
23471: \452_dbtreeview2access.txt \452_dbtrv3accessUML2.txt
23472: program TestDbTreeViewMainForm2_ACESS;
23473:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
23474:           +Exepath+'\examples\detail.mdb;Persist Security Info=False';
23475:   'Provider=MSDASQL.1;Persist Security Info=False;Extended
Properties="DSN=FB_EMPLOYEE;Driver=Firebird/InterBase(r)
driver;Dbname=C:\maXbox\maxbox3\mX3999\maxbox3\examples\EMPLOYEE.FDB;CHARSET=NONE;UID=SYSDBA;Role=Admin;"'
23476:
23477: OpenSSL Lib: unit ssl_openssl_lib;

```

```

23478: { $IFDEF CIL)
23479: const
23480:   ($IFDEF LINUX)
23481:     DLLSSLName = 'libssl.so';
23482:     DLLUtilName = 'libcrypto.so';
23483:   ($ELSE)
23484:     DLLSSLName = 'ssleay32.dll';
23485:     DLLUtilName = 'lbeay32.dll';
23486:   {$ENDIF}
23487: {$ELSE}
23488: var
23489:   ($IFNDEF MSWINDOWS)
23490:     ($IFDEF DARWIN)
23491:       DLLSSLName: string = 'libssl.dylib';
23492:       DLLUtilName: string = 'libcrypto.dylib';
23493:     ($ELSE)
23494:       DLLSSLName: string = 'libssl.so';
23495:       DLLUtilName: string = 'libcrypto.so';
23496:     {$ENDIF}
23497:   {$ELSE}
23498:     DLLSSLName: string = 'ssleay32.dll';
23499:     DLLSSLName2: string = 'libssl32.dll';
23500:     DLLUtilName: string = 'lbeay32.dll';
23501:   {$ENDIF}
23502: {$ENDIF}
23503:
23504: //-----
23505: //*****mX4 Macro Tags *****
23506: //-----
23507:
23508:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funcList399.txt end
23509:
23510: //Tag Macros in ini-file configure
23511:
23512:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
23513:
23514: //Tag Macros
23515: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
23516: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
23517: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
23518: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
23519: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
23520: 10199: SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
23521: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
23522: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
23523: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
  [getUserNameWin, getComputernameWin, datetimetoStr(now)],
23524: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
23525: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
23526: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
23527: 10199: [perftime numprocesssthreads, getIPAddress(getComputerNameWin),
23528: 10200: timetoStr(time), mbversion]),11);
23529: 10298: SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
23530: 10299: [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
23531:
23532:
23533:
23534: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
23535:
23536: //Replace Macros
23537:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
23538:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
23539:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
23540:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPATH, 9);
23541:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
23542:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
23543:   ref: netcologne.d1.sourceforge.net/project/maxbox/maxbox3.zip
23544:   SearchAndCopy(memo1.lines, '#tech'perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
23545: [perftime,numprocesssthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]], 11);
23546: //#tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
23547:   SearchAndCopy(memo1.lines, 'maxbox_extract_funcList399.txt
23548:   SearchAndCopy(memo1.lines, 'maxbox_extract_funcList422.txt
23549:
23550: //-----
23551: //*****mX4 ToDo List Tags ..../Help/ToDo List*****
23552: //-----
23553:
23554:   while I < sl.Count do begin
23555:     if MatchesMask(sl[I], '/? TODO ([a-z0-9_]*#[1-9#])*:*') then
23556:       if MatchesMask(sl[I], '/? TODO (?*#?#)*:*') then
23557:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
23558:       else if MatchesMask(sl[I], '/? DONE (?*#?#)*:*') then
23559:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
23560:       else if MatchesMask(sl[I], '/? TODO (#?#)*:*') then
23561:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
23562:       else if MatchesMask(sl[I], '/? DONE (#?#)*:*') then
23563:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
23564:       else if MatchesMask(sl[I], '/?*?TODO*:*)' then
23565:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO

```

```

23566:     else if MatchesMask(sl[I], '*/?*DONE*:*)' then
23567:       BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
23568:       Inc(I);
23569:   end;
23570;
23571: //-----
23572: //*****mX4 Public Tools API *****
23573: //-----
23574:   file : unit uPSI_fMain.pas;           [$OTAP] Open Tools API Catalog
23575: // Those functions concern the editor and preprocessor, all of the IDE
23576: Example: Call it with maxform1.InfoClick(self)
23577: Note: Call all Methods with maxForm1., e.g.:
23578:           maxForm1.ShellStyle1Click(self);
23579:
23580: procedure SIRegister_fMain(CL: TPSPascalCompiler);
23581: begin
23582:   Const('BYTECODE','String bytecode.txt'
23583:   Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT')
23584:   Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC'
23585:   Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS'
23586:   Const('PSINC','String PS Includes (*.inc)|*.INC'
23587:   Const('DEFFILENAME','String firstdemo.txt'
23588:   Const('DEFINIFILE','String 'maxboxdef.ini'
23589:   Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt'
23590:   Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt'
23591:   Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf'
23592:   Const('ALLOBJECTSLIST','String 'docs\VCL.pdf'
23593:   Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt'
23594:   Const('ALLUNITLIST','String 'docs\maxbox3_9.xml'
23595:   Const('INCLUDEBOX','String 'pas_includebox.inc'
23596:   Const('BOOTSCRIPT','String 'maxbootscript.txt'
23597:   Const('MBVERSION','String '4.2.4.80'
23598:   Const('VERSION','String'4.2.4.80
23599:   Const('MBVER','String '424
23600:   Const('MBVER1','Integer'(424);
23601:   Const('MBVERIAL1','Integer'(42480);
23602:   Const('EXENAME','String 'maXbox4.exe
23603:   Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm'
23604:   Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt'
23605:   Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt'
23606:   Const('MXINTERNETCHECK','String 'www.ask.com'
23607:   Const('MXMAIL','String 'max@kleiner.com'
23608:   Const('TAB','Char #$09';
23609:   Const('CODECOMPLETION','String 'bds_delphi.dci
23610:   SIRegister_TMaxForm1(CL);
23611: end;
23612;
23613: with FindClass('TForm'),'TMaxForm1') do begin
23614:   memo2', 'TMemo', iptrw);
23615:   memo1', 'TSynMemo', iptrw);
23616:   CB1SCList', 'TComboBox', iptrw);
23617:   mxNavigator', 'TComboBox', iptrw);
23618:   IPHost', 'string', iptrw);
23619:   IPPort', 'integer', iptrw);
23620:   COMPort', 'integer', iptrw); //3.9.6.4
23621:   Splitter1', 'TSplitter', iptrw);
23622:   PSScript', 'TPSScript', iptrw);
23623:   PS3D11Plugin', 'TPSD11Plugin', iptrw);
23624:   MainMen1', 'TMainMenu', iptrw);
23625:   Program1', 'TMenuItem', iptrw);
23626:   Compile1', 'TMenuItem', iptrw);
23627:   Files1', 'TMenuItem', iptrw);
23628:   open1', 'TMenuItem', iptrw);
23629:   Save2', 'TMenuItem', iptrw);
23630:   Options1', 'TMenuItem', iptrw);
23631:   Savebefore1', 'TMenuItem', iptrw);
23632:   Largefont1', 'TMenuItem', iptrw);
23633:   sBytecode1', 'TMenuItem', iptrw);
23634:   Saveas3', 'TMenuItem', iptrw);
23635:   Clear1', 'TMenuItem', iptrw);
23636:   Slinenumbers1', 'TMenuItem', iptrw);
23637:   About1', 'TMenuItem', iptrw);
23638:   Search1', 'TMenuItem', iptrw);
23639:   SynPasSyn1', 'TSynPasSyn', iptrw);
23640:   memo1', 'TSynMemo', iptrw);
23641:   SynEditSearch1', 'TSynEditSearch', iptrw);
23642:   WordWrap1', 'TMenuItem', iptrw);
23643:   XPMManifest1', 'TXPMManifest', iptrw);
23644:   SearchNext1', 'TMenuItem', iptrw);
23645:   Replace1', 'TMenuItem', iptrw);
23646:   PSImport_Controls1', 'TPSImport_Controls', iptrw);
23647:   PSImport_Classes1', 'TPSImport_Classes', iptrw);
23648:   ShowInclude1', 'TMenuItem', iptrw);
23649:   SynEditPrint1', 'TSynEditPrint', iptrw);
23650:   Printout1', 'TMenuItem', iptrw);
23651:   mnPrintColors1', 'TMenuItem', iptrw);
23652:   dlgFilePrint1', 'TPrintDialog', iptrw);
23653:   dlgPrintFont1', 'TFontDialog', iptrw);
23654:   mnuPrintFont1', 'TMenuItem', iptrw);

```

```

23655:     Include1', 'TMenuItem', iptrw);
23656:     CodeCompletionList1', 'TMenuItem', iptrw);
23657:     IncludeList1', 'TMenuItem', iptrw);
23658:     ImageList1', 'TImageList', iptrw);
23659:     ImageList2', 'TImageList', iptrw);
23660:     CoolBar1', 'TCoolBar', iptrw);
23661:     ToolBar1', 'TToolBar', iptrw);
23662:     btnLoad', 'TToolButton', iptrw);
23663:     ToolButton2', 'TToolButton', iptrw);
23664:     btnFind', 'TToolButton', iptrw);
23665:     btnCompile', 'TToolButton', iptrw);
23666:     btnTrans', 'TToolButton', iptrw);
23667:     btnUseCase', 'TToolButton', iptrw); //3.8
23668:     toolbtnTutorial', 'TToolButton', iptrw);
23669:     btn6res', 'TToolButton', iptrw);
23670:     ToolButton5', 'TToolButton', iptrw); 'ToolButton1', 'TToolButton', iptrw);
23671:     ToolButton3', 'TToolButton', iptrw); 'statusBar1', 'TStatusBar', iptrw);
23672:     SaveOutput1', 'TMenuItem', iptrw); 'ExportClipboard1', 'TMenuItem', iptrw);
23673:     Close1', 'TMenuItem', iptrw); 'Manual1', 'TMenuItem', iptrw);
23674:     About2', 'TMenuItem', iptrw); 'loadLastfile1', 'TMenuItem', iptrw);
23675:     imgLogo', 'TImage', iptrw); 'cedebug', 'TPSScriptDebugger', iptrw);
23676:     debugPopupMenu1', 'TPopupMenu', iptrw);
23677:     BreakPointMenu1', 'TMenuItem', iptrw);
23678:     Decompile1', 'TMenuItem', iptrw);
23679:     StepInto1', 'TMenuItem', iptrw);
23680:     StepOut1', 'TMenuItem', iptrw);
23681:     Reset1', 'TMenuItem', iptrw);
23682:     DebugRun1', 'TMenuItem', iptrw);
23683:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
23684:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
23685:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
23686:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
23687:     tutorial4', 'TMenuItem', iptrw);
23688:     ExporttoClipboard1', 'TMenuItem', iptrw);
23689:     ImportfromClipboard1', 'TMenuItem', iptrw);
23690:     N4', 'TMenuItem', iptrw); N5', 'TMenuItem', iptrw); N6', 'TMenuItem', iptrw);
23691:     ImportfromClipboard2', 'TMenuItem', iptrw);
23692:     tutorial1', 'TMenuItem', iptrw);
23693:     N7', 'TMenuItem', iptrw);
23694:     ShowSpecChar1', 'TMenuItem', iptrw);
23695:     OpenDirectory1', 'TMenuItem', iptrw);
23696:     procMess', 'TMenuItem', iptrw);
23697:     btnUseCase', 'TToolButton', iptrw);
23698:     ToolButton7', 'TToolButton', iptrw);
23699:     EditFont1', 'TMenuItem', iptrw);
23700:     UseCase1', 'TMenuItem', iptrw);
23701:     tutorial21', 'TMenuItem', iptrw);
23702:     OpenUseCase1', 'TMenuItem', iptrw);
23703:     PSImport_DB1', 'TPSImport_DB', iptrw);
23704:     tutorial31', 'TMenuItem', iptrw);
23705:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
23706:     HTMLSyntax1', 'TMenuItem', iptrw);
23707:     ShowInterfaces1', 'TMenuItem', iptrw);
23708:     Tutorial5', 'TMenuItem', iptrw);
23709:     AllFunctionsList1', 'TMenuItem', iptrw);
23710:     ShowLastException1', 'TMenuItem', iptrw);
23711:     PlayMP31', 'TMenuItem', iptrw);
23712:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
23713:     texSyntax1', 'TMenuItem', iptrw);
23714:     N8', 'TMenuItem', iptrw);
23715:     GetEMails1', 'TMenuItem', iptrw);
23716:     SynCppSyn1', 'TSynCppSyn', iptrw);
23717:     CSyntax1', 'TMenuItem', iptrw);
23718:     Tutorial6', 'TMenuItem', iptrw);
23719:     New1', 'TMenuItem', iptrw);
23720:     AllObjectsList1', 'TMenuItem', iptrw);
23721:     LoadBytecode1', 'TMenuItem', iptrw);
23722:     CipherFile1', 'TMenuItem', iptrw);
23723:     N9', 'TMenuItem', iptrw); N10', 'TMenuItem', iptrw);
23724:     Tutorial11', 'TMenuItem', iptrw);
23725:     Tutorial71', 'TMenuItem', iptrw);
23726:     UpdateService1', 'TMenuItem', iptrw);
23727:     PascalSchool1', 'TMenuItem', iptrw);
23728:     Tutorial81', 'TMenuItem', iptrw);
23729:     DelphiSite1', 'TMenuItem', iptrw);
23730:     Output1', 'TMenuItem', iptrw);
23731:     TerminalStyle1', 'TMenuItem', iptrw);
23732:     ReadOnly1', 'TMenuItem', iptrw);
23733:     ShellStyle1', 'TMenuItem', iptrw);
23734:     BigScreen1', 'TMenuItem', iptrw);
23735:     Tutorial91', 'TMenuItem', iptrw);
23736:     SaveOutput2', 'TMenuItem', iptrw);
23737:     N11', 'TMenuItem', iptrw);
23738:     SaveScreenshot', 'TMenuItem', iptrw);
23739:     Tutorial101', 'TMenuItem', iptrw);
23740:     SQLSyntax1', 'TMenuItem', iptrw);
23741:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
23742:     Console1', 'TMenuItem', iptrw);
23743:     SynXMLSyn1', 'TSynXMLSyn', iptrw);

```

```
23744: XMLSyntax1', 'TMenuItem', iptrw);
23745: ComponentCount1', 'TMenuItem', iptrw);
23746: NewInstance1', 'TMenuItem', iptrw);
23747: toolbarTutorial', 'TToolButton', iptrw);
23748: Memory1', 'TMenuItem', iptrw);
23749: SynJavaSyn1', 'TSynJavaSyn', iptrw);
23750: JavaSyntax1', 'TMenuItem', iptrw);
23751: SyntaxCheck1', 'TMenuItem', iptrw);
23752: Tutorial10Statistics1', 'TMenuItem', iptrw);
23753: ScriptExplorer1', 'TMenuItem', iptrw);
23754: FormOutput1', 'TMenuItem', iptrw);
23755: ArduinoDump1', 'TMenuItem', iptrw);
23756: AndroidDump1', 'TMenuItem', iptrw);
23757: GotoEnd1', 'TMenuItem', iptrw);
23758: AllResourceList1', 'TMenuItem', iptrw);
23759: ToolButton4', 'TToolButton', iptrw);
23760: btn6res', 'TToolButton', iptrw);
23761: Tutorial11Forms1', 'TMenuItem', iptrw);
23762: Tutorial12SQL1', 'TMenuItem', iptrw);
23763: ResourceExplore1', 'TMenuItem', iptrw);
23764: Info1', 'TMenuItem', iptrw);
23765: N12', 'TMenuItem', iptrw);
23766: CryptoBox1', 'TMenuItem', iptrw);
23767: Tutorial13Ciphering1', 'TMenuItem', iptrw);
23768: CipherFile2', 'TMenuItem', iptrw);
23769: N13', 'TMenuItem', iptrw);
23770: ModulesCount1', 'TMenuItem', iptrw);
23771: AddOns2', 'TMenuItem', iptrw);
23772: N4GewinntGame1', 'TMenuItem', iptrw);
23773: DocuforAddOns1', 'TMenuItem', iptrw);
23774: Tutorial14Async1', 'TMenuItem', iptrw);
23775: Lessons15Review1', 'TMenuItem', iptrw);
23776: SynPHPSyn1', 'TSynPHPSyn', iptrw);
23777: PHPSyntax1', 'TMenuItem', iptrw);
23778: Breakpoint1', 'TMenuItem', iptrw);
23779: SerialRS2321', 'TMenuItem', iptrw);
23780: N14', 'TMenuItem', iptrw);
23781: SynCSSyn1', 'TSynCSSyn', iptrw);
23782: CSyntax2', 'TMenuItem', iptrw);
23783: Calculator1', 'TMenuItem', iptrw);
23784: btnSerial', 'TToolButton', iptrw);
23785: ToolButton8', 'TToolButton', iptrw);
23786: Tutorial151', 'TMenuItem', iptrw);
23787: N15', 'TMenuItem', iptrw);
23788: N16', 'TMenuItem', iptrw);
23789: ControlBar1', 'TControlBar', iptrw);
23790: ToolBar2', 'TToolBar', iptrw);
23791: BtnOpen', 'TToolButton', iptrw);
23792: BtnSave', 'TToolButton', iptrw);
23793: BtnPrint', 'TToolButton', iptrw);
23794: BtnColors', 'TToolButton', iptrw);
23795: btnClassReport', 'TToolButton', iptrw);
23796: BtnRotateRight', 'TToolButton', iptrw);
23797: BtnFullScreen', 'TToolButton', iptrw);
23798: BtnFitToWindowSize', 'TToolButton', iptrw);
23799: BtnZoomMinus', 'TToolButton', iptrw);
23800: BtnZoomPlus', 'TToolButton', iptrw);
23801: Panel1', 'TPanel', iptrw);
23802: LabelBrettgroesse', ' TLabel', iptrw);
23803: CB1SCList1', 'TComboBox', iptrw);
23804: ImageListNormal', 'TImageList', iptrw);
23805: spbtnexpose', 'TSpeedButton', iptrw);
23806: spbtnexsample', 'TSpeedButton', iptrw);
23807: spbsaveas', 'TSpeedButton', iptrw);
23808: imglogobox', 'TImage', iptrw);
23809: EnlargeFont1', 'TMenuItem', iptrw);
23810: EnlargeFont2', 'TMenuItem', iptrw);
23811: ShrinkFont1', 'TMenuItem', iptrw);
23812: ThreadDemo1', 'TMenuItem', iptrw);
23813: HEXEditor1', 'TMenuItem', iptrw);
23814: HEXView1', 'TMenuItem', iptrw);
23815: HEXInspect1', 'TMenuItem', iptrw);
23816: SynExporterHTML1', 'TSynExporterHTML', iptrw);
23817: ExporttoHTML1', 'TMenuItem', iptrw);
23818: ClassCount1', 'TMenuItem', iptrw);
23819: HTMLOutput1', 'TMenuItem', iptrw);
23820: HEXEditor2', 'TMenuItem', iptrw);
23821: Minesweeper1', 'TMenuItem', iptrw);
23822: N17', 'TMenuItem', iptrw);
23823: PicturePuzzle1', 'TMenuItem', iptrw);
23824: sbvc1help', 'TSpeedButton', iptrw);
23825: DependencyWalker1', 'TMenuItem', iptrw);
23826: WebScanner1', 'TMenuItem', iptrw);
23827: View1', 'TMenuItem', iptrw);
23828: mnToolbar1', 'TMenuItem', iptrw);
23829: mnStatusbar2', 'TMenuItem', iptrw);
23830: mnConsole2', 'TMenuItem', iptrw);
23831: mnCoolbar2', 'TMenuItem', iptrw);
23832: mnSplitter2', 'TMenuItem', iptrw);
```

```

23833: WebServer1', 'TMenuItem', iptrw);
23834: Tutorial17Server1', 'TMenuItem', iptrw);
23835: Tutorial18Arduino1', 'TMenuItem', iptrw);
23836: SynPerlSyn1', 'TSynPerlSyn', iptrw);
23837: PerlSyntax1', 'TMenuItem', iptrw);
23838: SynPythonSyn1', 'TSynPythonSyn', iptrw);
23839: PythonSyntax1', 'TMenuItem', iptrw);
23840: DMathLibrary1', 'TMenuItem', iptrw);
23841: IntfNavigator1', 'TMenuItem', iptrw);
23842: EnlargeFontConsole1', 'TMenuItem', iptrw);
23843: ShrinkFontConsole1', 'TMenuItem', iptrw);
23844: SetInterfaceList1', 'TMenuItem', iptrw);
23845: popintfList', 'TPopupMenu', iptrw);
23846: intfAdd1', 'TMenuItem', iptrw);
23847: intfDelete1', 'TMenuItem', iptrw);
23848: intfRefactor1', 'TMenuItem', iptrw);
23849: Defactor1', 'TMenuItem', iptrw);
23850: Tutorial19COMArduino1', 'TMenuItem', iptrw);
23851: Tutorial20Regex', 'TMenuItem', iptrw);
23852: N18', 'TMenuItem', iptrw);
23853: ManualE1', 'TMenuItem', iptrw);
23854: FullTextFinder1', 'TMenuItem', iptrw);
23855: Move1', 'TMenuItem', iptrw);
23856: FractalDemo1', 'TMenuItem', iptrw);
23857: Tutorial21Android1', 'TMenuItem', iptrw);
23858: Tutorial0Function1', 'TMenuItem', iptrw);
23859: SimuLogBox1', 'TMenuItem', iptrw);
23860: OpenExamples1', 'TMenuItem', iptrw);
23861: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
23862: JavaScriptSyntax1', 'TMenuItem', iptrw);
23863: Halt1', 'TMenuItem', iptrw);
23864: CodeSearch1', 'TMenuItem', iptrw);
23865: SynRubySyn1', 'TSynRubySyn', iptrw);
23866: RubySyntax1', 'TMenuItem', iptrw);
23867: Undo1', 'TMenuItem', iptrw);
23868: SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
23869: LinuxShellScript1', 'TMenuItem', iptrw);
23870: Rename1', 'TMenuItem', iptrw);
23871: spdcodesearch', 'TSpeedButton', iptrw);
23872: Preview1', 'TMenuItem', iptrw);
23873: Tutorial22Services1', 'TMenuItem', iptrw);
23874: Tutorial23RealTime1', 'TMenuItem', iptrw);
23875: Configuration1', 'TMenuItem', iptrw);
23876: MP3Player1', 'TMenuItem', iptrw);
23877: DLLSpy1', 'TMenuItem', iptrw);
23878: SynURIOpener1', 'TSynURIOpener', iptrw);
23879: SynURISyn1', 'TSynURISyn', iptrw);
23880: URILinksClicks1', 'TMenuItem', iptrw);
23881: EditReplace1', 'TMenuItem', iptrw);
23882: GotoLine1', 'TMenuItem', iptrw);
23883: ActiveLineColor1', 'TMenuItem', iptrw);
23884: ConfigFile1', 'TMenuItem', iptrw);
23885: Sort1Intlist', 'TMenuItem', iptrw);
23886: Redo1', 'TMenuItem', iptrw);
23887: Tutorial24CleanCode1', 'TMenuItem', iptrw);
23888: Tutorial25Configuration1', 'TMenuItem', iptrw);
23889: IndentSelection1', 'TMenuItem', iptrw);
23890: UnindentSection1', 'TMenuItem', iptrw);
23891: SkyStyle1', 'TMenuItem', iptrw);
23892: N19', 'TMenuItem', iptrw);
23893: CountWords1', 'TMenuItem', iptrw);
23894: imbookmarkimages', 'TImageList', iptrw);
23895: Bookmark11', 'TMenuItem', iptrw);
23896: N20', 'TMenuItem', iptrw);
23897: Bookmark21', 'TMenuItem', iptrw);
23898: Bookmark31', 'TMenuItem', iptrw);
23899: Bookmark41', 'TMenuItem', iptrw);
23900: SynMultiSyn1', 'TSynMultiSyn', iptrw);
23901:
23902: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSCompiler)
23903: Procedure IFPS3ClassesPlugin1ExecImport(Sender:TObject;Exec:TPSEexec;x:TPSRuntimeClassImporter);
23904: Procedure PSScriptCompile( Sender : TPSScript)
23905: Procedure Compile1Click( Sender : TObject)
23906: Procedure PSScriptExecute( Sender : TPSScript)
23907: Procedure open1Click( Sender : TObject)
23908: Procedure Save2Click( Sender : TObject)
23909: Procedure Savebefore1Click( Sender : TObject)
23910: Procedure Largefont1Click( Sender : TObject)
23911: Procedure FormActivate( Sender : TObject)
23912: Procedure SBytecode1Click( Sender : TObject)
23913: Procedure FormKeyPress( Sender : TObject; var Key : Char)
23914: Procedure Saveas3Click( Sender : TObject)
23915: Procedure Clear1Click( Sender : TObject)
23916: Procedure Slinenumbers1Click( Sender : TObject)
23917: Procedure About1Click( Sender : TObject)
23918: Procedure Search1Click( Sender : TObject)
23919: Procedure FormCreate( Sender : TObject)
23920: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;
23921: var Action : TSynReplaceAction)

```

```

23922: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
23923: Procedure WordWrap1Click( Sender : TObject)
23924: Procedure SearchNext1Click( Sender : TObject)
23925: Procedure Replace1Click( Sender : TObject)
23926: Function PSScriptNeedFile(Sdr:TObject;const OriginFileName:Str;var FName,Output:Str):Bool;
23927: Procedure ShowInclude1Click( Sender : TObject)
23928: Procedure Printout1Click( Sender : TObject)
23929: Procedure mnuPrintFont1Click( Sender : TObject)
23930: Procedure Include1Click( Sender : TObject)
23931: Procedure FormDestroy( Sender : TObject)
23932: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
23933: Procedure UpdateView1Click( Sender : TObject)
23934: Procedure CodeCompletionList1Click( Sender : TObject)
23935: Procedure SaveOutput1Click( Sender : TObject)
23936: Procedure ExportClipboard1Click( Sender : TObject)
23937: Procedure Close1Click( Sender : TObject)
23938: Procedure Manual1Click( Sender : TObject)
23939: Procedure LoadLastFile1Click( Sender : TObject)
23940: Procedure Memo1Change( Sender : TObject)
23941: Procedure Decompile1Click( Sender : TObject)
23942: Procedure StepInto1Click( Sender : TObject)
23943: Procedure StepOut1Click( Sender : TObject)
23944: Procedure Reset1Click( Sender : TObject)
23945: Procedure cedebugAfterExecute( Sender : TPSScript)
23946: Procedure cedebugBreakpoint(Sender:TObject;const FileName:String;Position,Row,Col:Cardinal)
23947: Procedure cedebugCompile( Sender : TPSScript)
23948: Procedure cedebugExecute( Sender : TPSScript)
23949: Procedure cedebugIdle( Sender : TObject)
23950: Procedure cedebugLineInfo(Sender:TObject;const FileName:String;Position,Row,Col: Cardinal)
23951: Procedure Memo1SpecialLineColor(Sender:TObject;Line:Int;var Speci:Bool;var FG,BG:TColor);
23952: Procedure BreakPointMenuClick( Sender : TObject)
23953: Procedure DebugRun1Click( Sender : TObject)
23954: Procedure tutorial4Click( Sender : TObject)
23955: Procedure ImportfromClipboard1Click( Sender : TObject)
23956: Procedure ImportfromClipboard2Click( Sender : TObject)
23957: Procedure tutorial1Click( Sender : TObject)
23958: Procedure ShowSpecChars1Click( Sender : TObject)
23959: Procedure StatusBar1DblClick( Sender : TObject)
23960: Procedure PSScriptLine( Sender : TObject)
23961: Procedure OpenDirectory1Click( Sender : TObject)
23962: Procedure procMessClick( Sender : TObject)
23963: Procedure btnUseCaseClick( Sender : TObject)
23964: Procedure EditFont1Click( Sender : TObject)
23965: Procedure tutorial21Click( Sender : TObject)
23966: Procedure tutorial31Click( Sender : TObject)
23967: Procedure HTMLSyntax1Click( Sender : TObject)
23968: Procedure ShowInterfaces1Click( Sender : TObject)
23969: Procedure Tutorial5Click( Sender : TObject)
23970: Procedure ShowLastException1Click( Sender : TObject)
23971: Procedure PlayMP31Click( Sender : TObject)
23972: Procedure AllFunctionsList1Click( Sender : TObject)
23973: Procedure texSyntax1Click( Sender : TObject)
23974: Procedure GetEMails1Click( Sender : TObject)
23975: procedure DelphiSite1Click(Sender: TObject);
23976: procedure TerminalStyle1Click(Sender: TObject);
23977: procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
23978: procedure ShellStyle1Click(Sender: TObject);
23979: procedure Console1Click(Sender: TObject); //3.2
23980: procedure BigScreen1Click(Sender: TObject);
23981: procedure Tutorial91Click(Sender: TObject);
23982: procedure SaveScreenshotClick(Sender: TObject);
23983: procedure Tutorial101Click(Sender: TObject);
23984: procedure SQLSyntax1Click(Sender: TObject);
23985: procedure XMLSyntax1Click(Sender: TObject);
23986: procedure ComponentCount1Click(Sender: TObject);
23987: procedure NewInstance1Click(Sender: TObject);
23988: procedure CSyntax1Click(Sender: TObject);
23989: procedure Tutorial6Click(Sender: TObject);
23990: procedure New1Click(Sender: TObject);
23991: procedure AllObjectsList1Click(Sender: TObject);
23992: procedure LoadBytecode1Click(Sender: TObject);
23993: procedure CipherFile1Click(Sender: TObject); //V3.5
23994: procedure NewInstance1Click(Sender: TObject);
23995: procedure toolbarTutorialclick(Sender: TObject);
23996: procedure Memory1Click(Sender: TObject);
23997: procedure JavaSyntax1Click(Sender: TObject);
23998: procedure SyntaxCheck1Click(Sender: TObject);
23999: procedure ScriptExplorer1Click(Sender: TObject);
24000: procedure FormOutput1Click(Sender: TObject); //V3.6
24001: procedure GotoEnd1Click(Sender: TObject);
24002: procedure AllResourceList1Click(Sender: TObject);
24003: procedure btn6resClick(Sender: TObject); //V3.7
24004: procedure Info1Click(Sender: TObject);
24005: procedure Tutorial10Statistics1Click(Sender: TObject);
24006: procedure Tutorial11Forms1Click(Sender: TObject);
24007: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
24008: procedure ResourceExplore1Click(Sender: TObject);
24009: procedure Info1Click(Sender: TObject);
24010: procedure CryptoBox1Click(Sender: TObject);

```

```

24011: procedure ModulesCount1Click(Sender: TObject);
24012: procedure N4GewinntGame1Click(Sender: TObject);
24013: procedure PHPSyntax1Click(Sender: TObject);
24014: procedure SerialRS2321Click(Sender: TObject);
24015: procedure CSyntax2Click(Sender: TObject);
24016: procedure Calculator1Click(Sender: TObject);
24017: procedure Tutorial13Ciphering1Click(Sender: TObject);
24018: procedure Tutorial14Async1Click(Sender: TObject);
24019: procedure PHPSyntax1Click(Sender: TObject);
24020: procedure BtnZoomPlusClick(Sender: TObject);
24021: procedure BtnZoomMinusClick(Sender: TObject);
24022: procedure btnClassReportClick(Sender: TObject);
24023: procedure ThreadDemo1Click(Sender: TObject);
24024: procedure HEXView1Click(Sender: TObject);
24025: procedure ExporttoHTML1Click(Sender: TObject);
24026: procedure Minesweeper1Click(Sender: TObject);
24027: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
24028: procedure sbvc1helpClick(Sender: TObject);
24029: procedure DependencyWalker1Click(Sender: TObject);
24030: procedure CB1SCListDrawItem(Control:TWinCtrl;Index:Int;aRect:TRect;State:TOwnerDrawState);
24031: procedure WebScanner1Click(Sender: TObject);
24032: procedure mnToolbar1Click(Sender: TObject);
24033: procedure mnStatusbar2Click(Sender: TObject);
24034: procedure mnConsole2Click(Sender: TObject);
24035: procedure mnCoolbar2Click(Sender: TObject);
24036: procedure mnSplitter2Click(Sender: TObject);
24037: procedure WebServer1Click(Sender: TObject);
24038: procedure PerlSyntax1Click(Sender: TObject);
24039: procedure PythonSyntax1Click(Sender: TObject);
24040: procedure DMathLibrary1Click(Sender: TObject);
24041: procedure IntfNavigator1Click(Sender: TObject);
24042: procedure FullTextFinder1Click(Sender: TObject);
24043: function AppName: string;
24044: function ScriptName: string;
24045: function LastName: string;
24046: procedure FractalDemo1Click(Sender: TObject);
24047: procedure SimuLogBox1Click(Sender: TObject);
24048: procedure OpenExamples1Click(Sender: TObject);
24049: procedure Halt1Click(Sender: TObject);
24050: procedure Stop;
24051: procedure CodeSearch1Click(Sender: TObject);
24052: procedure RubySyntax1Click(Sender: TObject);
24053: procedure Undo1Click(Sender: TObject);
24054: procedure LinuxShellScript1Click(Sender: TObject);
24055: procedure WebScannerDirect(urls: string);
24056: procedure WebScanner(urls: string);
24057: procedure LoadInterfaceList2;
24058: procedure DLLSpy1Click(Sender: TObject);
24059: procedure Memo1DblClick(Sender: TObject);
24060: procedure URILinksClicks1Click(Sender: TObject);
24061: procedure Gotoline1Click(Sender: TObject);
24062: procedure ConfigFile1Click(Sender: TObject);
24063: Procedure Sort1IntflistClick( Sender : TObject)
24064: Procedure Redo1Click( Sender : TObject)
24065: Procedure Tutorial24CleanCode1Click( Sender : TObject)
24066: Procedure IndentSelection1Click( Sender : TObject)
24067: Procedure UnindentSection1Click( Sender : TObject)
24068: Procedure SkyStyle1Click( Sender : TObject)
24069: Procedure CountWords1Click( Sender : TObject)
24070: Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
24071: Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
24072: Procedure Bookmark11Click( Sender : TObject)
24073: Procedure Bookmark21Click( Sender : TObject)
24074: Procedure Bookmark31Click( Sender : TObject)
24075: Procedure Bookmark41Click( Sender : TObject)
24076: Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
24077: 'STATMemoryReport', 'boolean', iptrw);
24078: 'IPPort', 'integer', iptrw);
24079: 'COMPort', 'integer', iptrw);
24080: 'lbintflist', 'TListBox', iptrw);
24081: Function GetStatChange : boolean
24082: Procedure SetStatChange( vstat : boolean)
24083: Function GetActFileName : string
24084: Procedure SetActFileName( vname : string)
24085: Function GetLastFileName : string
24086: Procedure SetLastFileName( vname : string)
24087: Procedure WebScannerDirect( urls : string)
24088: Procedure LoadInterfaceList2
24089: Function GetStatExecuteShell : boolean
24090: Procedure DoEditorExecuteCommand( EditorCommand : word)
24091: function GetActiveLineColor: TColor
24092: procedure SetActiveLineColor(acolor: TColor)
24093: procedure ScriptListbox1Click(Sender: TObject);
24094: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
24095: procedure EnlargeGutter1Click(Sender: TObject);
24096: procedure Tetris1Click(Sender: TObject);
24097: procedure ToDoList1Click(Sender: TObject);
24098: procedure ProcessList1Click(Sender: TObject);
24099: procedure MetricReport1Click(Sender: TObject);

```

```

24100: procedure ProcessList1Click(Sender: TObject);
24101: procedure TCPSockets1Click(Sender: TObject);
24102: procedure ConfigUpdate1Click(Sender: TObject);
24103: procedure ADOWorkbench1Click(Sender: TObject);
24104: procedure SocketServer1Click(Sender: TObject);
24105: procedure FormDemo1Click(Sender: TObject);
24106: procedure Richedit1Click(Sender: TObject);
24107: procedure SimpleBrowser1Click(Sender: TObject);
24108: procedure DOSShell1Click(Sender: TObject);
24109: procedure SynExport1Click(Sender: TObject);
24110: procedure ExporttoRTF1Click(Sender: TObject);
24111: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
24112: procedure SOAPTester1Click(Sender: TObject);
24113: procedure Sniffer1Click(Sender: TObject);
24114: procedure AutoDetectSyntax1Click(Sender: TObject);
24115: procedure FPPlot1Click(Sender: TObject);
24116: procedure PassStyle1Click(Sender: TObject);
24117: procedure Tutorial183RGBLED1Click(Sender: TObject);
24118: procedure Reversi1Click(Sender: TObject);
24119: procedure ManualmaXbox1Click(Sender: TObject);
24120: procedure BlaisePascalMagazine1Click(Sender: TObject);
24121: procedure AddToDo1Click(Sender: TObject);
24122: procedure CreateGUID1Click(Sender: TObject);
24123: procedure Tutorial27XML1Click(Sender: TObject);
24124: procedure CreateDLLStub1Click(Sender: TObject);
24125: procedure Tutorial28DLL1Click(Sender: TObject);');
24126: procedure ResetKeyPressed;');
24127: procedure SetKeyPressed;');
24128: procedure KeyPressedFalse;
24129: procedure FileChanges1Click(Sender: TObject);');
24130: procedure OpenGLTry1Click(Sender: TObject);';
24131: procedure AllUnitList1Click(Sender: TObject);';
24132: procedure Tutorial29UMLClick(Sender: TObject);
24133: procedure CreateHeader1Click(Sender: TObject);
24134: procedure Oscilloscope1Click(Sender: TObject);';
24135: procedure Tutorial30WOT1Click(Sender: TObject);';
24136: procedure GetWebScript1Click(Sender: TObject);';
24137: procedure Checkers1Click(Sender: TObject);';
24138: procedure TaskMgr1Click(Sender: TObject);';
24139: procedure WebCam1Click(Sender: TObject);';
24140: procedure Tutorial31Closure1Click(Sender: TObject);';
24141: procedure GEOFMapView1Click(Sender: TObject);';
24142: procedure Run1Click(Sender: TObject);
24143: MaxForm1.GPSSatView1Click, 'GPSSatView1Click');
24144: MaxForm1.N3DLabi1Click, 'N3DLabi1Click');
24145: procedure ExternalApp1Click(Sender: TObject);';
24146: procedure PANView1Click(Sender: TObject);
24147: procedure Tutorial39GEOMaps1Click(Sender: TObject);
24148: procedure UnitConverter1Click(Sender: TObject);
24149: maxform1.myscript1click(self);
24150: procedure Terminal1Click(Sender: TObject);
24151: procedure Tutorial361Click(Sender: TObject);
24152: procedure TrainingArduino1Click(Sender: TObject);
24153: procedure Chess41Click(Sender: TObject);
24154: procedure OrangeStyle1Click(Sender: TObject);
24155: //-----
24156: //*****mX4 Editor SynEdit Tools API *****
24157: //-----
24158: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
24159: begin //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
24160: with FindClass ('TCustomControl'),'TCustomSynEdit') do begin
24161:   Constructor Create( AOwner : TComponent)
24162:   SelStart', 'Integer', iptrw';
24163:   SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
24164:   Procedure UpdateCaret
24165:   Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
24166:   Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
24167:   Procedure BeginUndoBlock
24168:   Procedure BeginUpdate
24169:   Function CaretInView : Boolean
24170:   Function CharIndexToRowCol( Index : integer) : TBufferCoord
24171:   Procedure Clear
24172:   Procedure ClearAll
24173:   Procedure ClearBookMark( BookMark : Integer)
24174:   Procedure ClearSelection
24175:   Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
24176:   Procedure ClearUndo
24177:   Procedure CopyToClipboard
24178:   Procedure CutToClipboard
24179:   Procedure DoCopyToClipboard( const SText : string)
24180:   Procedure EndUndoBlock
24181:   Procedure EndUpdate
24182:   Procedure EnsureCursorPosVisible
24183:   Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
24184:   Procedure FindMatchingBracket
24185:   Function GetMatchingBracket : TBufferCoord
24186:   Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
24187:   Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
24188:   Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean

```

```

24189: Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attr
24190:           : TSynHighlighterAttributes) : boolean
24191: Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
24192:           var TokenType, Start : Integer; var Attr:TSynHighlighterAttributes):boolean
24193: Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
24194: Function GetWordAtRowCol( const XY : TBufferCoord) : string
24195: Procedure GotoBookMark( BookMark : Integer)
24196: Procedure GotoLineAndCenter( ALine : Integer)
24197: Function IdentChars : TSynIdentChars
24198: Procedure InvalidateGutter
24199: Procedure InvalidateGutterLine( aLine : integer)
24200: Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
24201: Procedure InvalidateLine( Line : integer)
24202: Procedure InvalidateLines( FirstLine, LastLine : integer)
24203: Procedure InvalidateSelection
24204: Function IsBookmark( BookMark : integer) : boolean
24205: Function IsPointInSelection( const Value : TBufferCoord) : boolean
24206: Procedure LockUndo
24207: Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
24208: Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
24209: Function LineToRow( aLine : integer) : integer
24210: Function RowToLine( aRow : integer) : integer
24211: Function NextWordPos : TBufferCoord
24212: Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
24213: Procedure PasteFromClipboard
24214: Function WordStart : TBufferCoord
24215: Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
24216: Function WordEnd : TBufferCoord
24217: Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
24218: Function PrevWordPos : TBufferCoord
24219: Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
24220: Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
24221: Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
24222: Procedure Redo
24223: Procedure RegisterCommandHandler( const AHandlerProc:THookedCommandEvent; AHandlerData:pointer);
24224: Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
24225: Function RowColToCharIndex( RowCol : TBufferCoord) : integer
24226: Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): int
24227: Procedure SelectAll
24228: Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
24229: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
24230: Procedure SetDefaultKeystrokes
24231: Procedure SetSelWord
24232: Procedure SetWordBlock( Value : TBufferCoord)
24233: Procedure Undo Procedure UnlockUndo
24234: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
24235: Procedure AddKeyUpHandler( aHandler : TKeyEvent)
24236: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
24237: Procedure AddKeyDownHandler( aHandler : TKeyEvent)
24238: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
24239: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
24240: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
24241: Procedure AddFocusControl( aControl : TWInControl)
24242: Procedure RemoveFocusControl( aControl : TWInControl)
24243: Procedure AddMouseDownHandler( aHandler : TMouseEvent)
24244: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
24245: Procedure AddMouseUpHandler( aHandler : TMouseEvent)
24246: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
24247: Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
24248: Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
24249: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
24250: Procedure RemoveLinesPointer
24251: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
24252: Procedure UnHookTextBuffer
24253: BlockBegin', 'TBufferCoord', iptrw';
24254: BlockEnd', 'TBufferCoord', iptrw';
24255: CanPaste', 'Boolean', iptr); 'Canredo', 'boolean', iptr);
24256: Canundo', 'boolean', iptr); 'CaretX', 'Integer', iptrw);
24257: CaretY', 'Integer', iptrw); 'CaretXY', 'TBufferCoord', iptrw);
24258: ActiveLineColor', 'TColor', iptrw);
24259: DisplayX', 'Integer', iptr); 'DisplayY', 'Integer', iptr);
24260: DisplayXY', 'TDisplayCoord', iptr); 'DisplayLineCount', 'integer', iptr);
24261: CharsInWindow', 'Integer', iptr);
24262: CharWidth', 'integer', iptr);
24263: Font', 'TFont', iptrw);
24264: GutterWidth', 'Integer', iptr);
24265: Highlighter', 'TSynCustomHighlighter', iptrw);
24266: LeftChar', 'Integer', iptrw);
24267: LineHeight', 'integer', iptr);
24268: LinesInWindow', 'Integer', iptr);
24269: LineText', 'string', iptrw); Lines', 'TStrings', iptrw);
24270: Marks', 'TSynEditMarkList', iptr);
24271: MaxScrollWidth', 'integer', iptrw);
24272: Modified', 'Boolean', iptrw);
24273: PaintLock', 'Integer', iptr);
24274: ReadOnly', 'Boolean', iptrw);
24275: SearchEngine', 'TSynEditSearchCustom', iptrw);
24276: SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
24277: SelTabBlock', 'Boolean', iptr);

```

```

24278:     SelTabLine', 'Boolean', iptr); SelText', 'string', iptrw);
24279:     StateFlags', 'TSynStateFlags', iptr);
24280:     Text', 'string', iptrw); TopLine', 'Integer', iptrw);
24281:     WordAtCursor', 'string', iptr); WordAtMouse', 'string', iptr);
24282:     UndoList', 'TSynEditUndoList', iptr);
24283:     RedoList', 'TSynEditUndoList', iptr);
24284:     OnProcessCommandEvent', 'TProcessEvent', iptrw);
24285:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
24286:     BorderStyle', 'TSynBorderStyle', iptrw);
24287:     ExtraLineSpacing', 'integer', iptrw);
24288:     Gutter', 'TSynGutter', iptrw);
24289:     HideSelection', 'boolean', iptrw);
24290:     InsertCaret', 'TSynEditCaretType', iptrw);
24291:     InsertMode', 'boolean', iptrw); IsScrolling', 'Boolean', iptr);
24292:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
24293:     MaxUndo', 'Integer', iptrw); Options', 'TSynEditorOptions', iptrw);
24294:     OverwriteCaret', 'TSynEditCaretType', iptrw);
24295:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
24296:     ScrollHintColor', 'TColor', iptrw);
24297:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
24298:     ScrollBars', 'TScrollStyle', iptrw);
24299:     SelectedColor', 'TSynSelectedColor', iptrw);
24300:     SelectionMode', 'TSynSelectionMode', iptrw);
24301:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
24302:     TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
24303:     WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
24304:     WordWrapGlyph', 'TSynGlyph', iptrw);
24305:     OnChange', 'TNotifyEvent', iptrw);
24306:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
24307:     OnCommandProcessed', 'TProcessEvent', iptrw);
24308:     OnContextHelp', 'TContextHelpEvent', iptrw);
24309:     OnDropFiles', 'TDropFilesEvent', iptrw);
24310:     OnGutterClick', 'TGutterClickEvent', iptrw);
24311:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
24312:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
24313:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
24314:     OnPaint', 'TPaintEvent', iptrw);
24315:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
24316:     OnProcessUserCommand', 'TProcessEvent', iptrw);
24317:     OnReplaceText', 'TReplaceTextEvent', iptrw);
24318:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
24319:     OnStatusChange', 'TStatusChangeEvent', iptrw);
24320:     OnPaintTransient', 'TPaintTransient', iptrw);
24321:     OnScroll', 'TScrollEvent', iptrw);
24322:   end;
24323: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
24324: Function GetPlaceableHighlighters : TSynHighlighterList
24325: Function EditorCommandToDescrString( Cmd : TSynEditorCommand ) : string
24326: Function EditorCommandToCodeString( Cmd : TSynEditorCommand ) : string
24327: Procedure GetEditorCommandValues( Proc : TGetStrProc )
24328: Procedure GetEditorCommandExtended( Proc : TGetStrProc )
24329: Function IdentToEditorCommand( const Ident : string; var Cmd : longint ) : boolean
24330: Function EditorCommandToIdent( Cmd : longint; var Ident : string ) : boolean
24331: Function ConvertCodeStringToExtended( AString : String ) : String
24332: Function ConvertExtendedToCodeString( AString : String ) : String
24333: Function ConvertExtendedToCommand( AString : String ) : TSynEditorCommand
24334: Function ConvertCodeStringToCommand( AString : String ) : TSynEditorCommand
24335: Function IndexToEditorCommand( const AIndex : Integer ) : Integer
24336:
24337: TSynEditorOption = (
24338:   eoAltSetsColumnMode,           //Holding down Alt Key will put select mode into columnar format
24339:   eoAutoIndent,                //Will indent caret on newlines same amount of leading whitespace as
24340:                           //preceding line
24341:   eoAutoSizeMaxScrollWidth,    //Automatically resizes MaxScrollWidth property when insert text
24342:   eoDisableScrollArrows,       //Disables scroll bar arrow buttons when you can't scroll that
24343:                           //direction any more
24344:   eoDragDropEditing,          //Allows to select a textblock and drag it to another location
24345:   eoDropFiles,                 //Allows the editor accept OLE file drops
24346:   eoEnhanceHomeKey,           //enhances home key positioning, similar to visual studio
24347:   eoEnhanceEndKey,            //enhances End key positioning, similar to JDeveloper
24348:   eoGroupUndo,                //When undoing/redoing actions, handle all changes same kind in one call
24349:                           //instead undoing/redoing each command separately
24350:   eoHalfPageScroll,           //By scrolling with page-up/page-down commands, scroll half page at time
24351:   eoHideShowScrollbars,        //if enabled, then scrollbars will only show if necessary.
24352:   If you have ScrollPastEOL,  then it the horizontal bar will always be there (uses MaxLength instead)
24353:   eoKeepCaretX,               //When moving through lines w/o cursor past EOL, keeps X pos of cursor
24354:   eoNoCaret,                  //Makes it so the caret is never visible
24355:   eoNoSelection,              //Disables selecting text
24356:   eoRightMouseMovesCursor,    //When clicking right mouse for popup menu, moves cursor to location
24357:   eoscrollByOneLess,          //Forces scrolling to be one less
24358:   eoscrollHintFollows,        //The scroll hint follows the mouse when scrolling vertically
24359:   eoscrollPastEof,            //Allows the cursor to go past the end of file marker
24360:   eoscrollPastEol,             //Allows cursor to go past last char into white space at end of line
24361:   eoshowScrollHint,           //Shows a hint of the visible line numbers when scrolling vertically
24362:   eoshowSpecialChars,         //Shows the special characters
24363:   eosmartTabDelete,           //similar to Smart Tabs, but when you delete characters
24364:   eosmartTabs,                //When tabbing, cursor will go to non-whitespace char of prev line
24365:   eospecialLineDefaultFg,     //disables foreground textcolor override using OnSpecialLineColor event
24366:   eotabIndent,                //If active <Tab> and <Shift><Tab> act block indent, unindent if text select

```

```

24367: eoTabsToSpaces,           //Converts a tab character to a specified number of space characters
24368: eoTrimTrailingSpaces   //Spaces at the end of lines will be trimmed and not saved
24369:
24370: *****Important Editor Short Cuts*****;
24371: Double click to select a word and count words with highlighting.
24372: Triple click to select a line.
24373: CTRL+SHIFT+click to extend a selection.
24374: Drag with the ALT key down to select columns of text !!!
24375: Drag and drop is supported.
24376: With CTRL 3 to jump to the last change (change tracker)
24377: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
24378: Type CTRL+A to select all.
24379: Type CTRL+N to set a new line.
24380: Type CTRL+T to delete a line or token. //Tokenizer
24381: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
24382: Type CTRL+Shift+T to add ToDo in line and list.
24383: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
24384: Type CTRL+[0..9] to jump or get to bookmarks.
24385: Type Home to position cursor at beginning of current line and End to position it at end of line.
24386: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
24387: Page Up and Page Down work as expected.
24388: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
24389: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
24390:
24391: {# Short Key Positions Ctrl<A-Z>: }
24392: def
24393: <A> Select All
24394: <B> Count Words
24395: <C> Copy
24396: <D> Internet Start
24397: <E> Script List
24398: <F> Find
24399: <G> Goto
24400: <H> Mark Line
24401: <I> Interface List
24402: <J> Code Completion
24403: <K> Console
24404: <L> Interface List Box
24405: <M> Font Smaller -
24406: <N> New Line
24407: <O> Open File
24408: <P> Font Larger +
24409: <Q> Quit
24410: <R> Replace
24411: <S> Save!
24412: <T> Delete Line
24413: <U> Use Case Editor
24414: <V> Paste
24415: <W> URI Links
24416: <X> Reserved for coding use internal
24417: <Y> Delete Line
24418: <Z> Undo
24419:
24420: ref F1 Help
24421: F2 Syntax Check
24422: F3 Search Next
24423: F4 New Instance
24424: F5 Line Mark /Breakpoint
24425: F6 Goto End
24426: F7 Debug Step Into
24427: F8 Debug Step Out
24428: F9 Compile
24429: F10 Menu
24430: F11 Word Count Highlight
24431: F12 Reserved for coding use internal
24432:
24433: AddRegisteredVariable('Application', 'TApplication');
24434: AddRegisteredVariable('Screen', 'TScreen');
24435: AddRegisteredVariable('Self', 'TForm');
24436: AddRegisteredVariable('Memo1', 'TSynMemo');
24437: AddRegisteredVariable('memo2', 'TMemo');
24438: AddRegisteredVariable('maxForm1', 'TMaxform1'); //!
24439: AddRegisteredVariable('debugout', 'Tdebugoutput'); //!
24440: AddRegisteredVariable('hlog', 'THotlog'); //!
24441: AddRegisteredVariable('mouse', 'TMouse'); //!
24442: AddRegisteredVariable( it ,integer); //for closure and each loop!
24443: AddRegisteredVariable( sr ,string); //for closure
24444: AddRegisteredVariable( bt ,boolean); //for closure
24445: AddRegisteredVariable( ft ,double); //for closure
24446: AddRegisteredVariable( srlist , TStringlist); //for closures
24447:
24448: def ReservedWords: array[0..86] of string =
24449: ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
24450: 'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
24451: 'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
24452: 'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
24453: 'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
24454: 'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
24455: 'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',

```

```

24456:     'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
24457:     'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
24458:     'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
24459:     'public', 'published', def, ref, using, typedef, memo1', 'memo2', 'doc', 'maxform1', 'it';
24460:     AllowedChars: array[0..5] of string = ('(', ')', '[', ']', ',', t, t1, t2, t3: boolean;
24461: //-----
24462: //*****End of mX4 Public Tools API ****
24463: //-----
24464:
24465: maXbox 4 Internal Now Inventory INI
24466:
24467: Amount of Functions: 17679
24468: Amount of Procedures: 10532
24469: Amount of Constructors: 1686
24470: Totals of Calls: 29897
24471: SHA1: of 4.2.4.80 15565A557B0F9576AA5F23F2A1D06BE9699A757B
24472: CRC32: FA1F1F25 26.4 MB (27,720,144 bytes)
24473: 000 mX4 executed: 14/10/2016 22:11:32 Runtime: 0:7:22.399 Memload: 23% use
24474: 000
24475: Amount of Functions: 17059
24476: Amount of Procedures: 10218
24477: Amount of Constructors: 1679
24478: Totals of Calls: 28956
24479: SHA1: of 4.2.4.25 A52ACF844808285D8EE978637365B74B3C7C342F
24480: CRC32: CB882FFC 26.0 MB (27,276,288 bytes)
24481: 000 mX4 executed: 05/06/2016 17:29:08 Runtime: 0:6:52.817 Memload: 25% use
24482:
24483: Amount of Functions: 16910
24484: Amount of Procedures: 10173
24485: Amount of Constructors: 1679
24486: Totals of Calls: 28762
24487: SHA1: of 4.2.2.95 3EBECE1B08602BBC2785B1C9B8827EECF092330
24488: CRC32: 21D1039C, Size of EXE: 27,179,008
24489:
24490: Amount of Functions: 16627
24491: Amount of Procedures: 9922
24492: Amount of Constructors: 1618
24493: Totals of Calls: 27862
24494: SHA1: of maXbox4.exe (4.0.2.80) CDC0D39FE16CE883EA98FF65C7E31C874FE1520B
24495: CRC32: 64E170B0
24496: maXbox4.exe 26,506,752 bytes
24497:
24498: Amount of Functions: 16507
24499: Amount of Procedures: 9872
24500: Amount of Constructors: 1627
24501: Totals of Calls: 28006
24502:
24503: Amount of Functions: 16627
24504: Amount of Procedures: 9922
24505: Amount of Constructors: 1633
24506: Totals of Calls: 28128
24507: SHA1: of maXbox 4.2.0.80 638E7412750AB0ECF14F2A5515BC4A2DE561EAC2
24508: CRC32: 7C91FD2A Exe size: 26,650,112
24509: https://www.virustotal.
com/en/file/584ca53d6dd8f7de17d0a1959bf78aad04697cf0ad7b3f23aee90b5ff720ede1/analysis/1460194451/
24510:
24511: Update for Windows 10 Version 1511 for x64-based Systems (KB3140741)
24512: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3140768)
24513: Amount of Functions: 16746
24514: Amount of Procedures: 10028
24515: Amount of Constructors: 1652
24516: Totals of Calls: 28426!
24517: SHA1: of 4.2.2.90 1EEE461ACF78ACA461806D85488B87A4FA08F39F
24518: CRC32: BCEDACF0 Size of EXE: 26,920,960
24519:
24520: ****
24521: Doc Short Manual with 50 Tips!
24522: ****
24523: - Install: just save your maxboxdef.ini before and then extract the zip file!
24524: - Toolbar: Click on the red maXbox Sign (right on top) opens your work dir or jump to <Help>
24525: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
24526: - Menu: With <Ctrl1><F3> you can search for code on examples
24527: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
24528: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
24529: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
24530:
24531: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
24532: - Inifile: Refresh (reload) the inifile after edit with ..../Help/Config Update
24533: - Context Menu: You can printout your scripts as a pdf-file or html-export
24534: - Context: You do have a context menu with the right mouse click
24535:
24536: - Menu: With the UseCase Editor you can convert graphic formats too.
24537: - Menu: On menu Options you find Addons as compiled scripts
24538: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
24539: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
24540: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
24541: - Editor: You can get templates as code completion with <ctrl j> in editor like classsp or iinterface or
ttimer (you type classsp and then CTRL J), or you type tstringlist and <Ctrl><J>
24542:

```

```

24543: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
24544: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
24545: - Code: If you code a loop till key-pressed use function: isKeyPressed;
24546: - Code: Macro set the macros #name,, #paAdministrorth, #file,startmaxbox_extract_funclist5.pdf
24547: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
24548: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
24549:     to delete and Click and mark to drag a bookmark
24550: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
24551: - IDE: A file info with system and script information you find in menu Program/Information
24552: - IDE: After change the config file in help you can update changes in menu Help/Config Update
24553: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
24554: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time start it.
24555: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
24556: - Editor: Set Bookmarks to check your work in app or code
24557: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
24558: - Editor: With //TODO: some description or DONE you set code entries for ToDo List in ..../Help/ToDo List
24559: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
24560: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
24561: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
24562: - Menu: Set Interface Naviagator also with toggle <Ctrl L> or /View/Intf Navigator
24563: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
24564: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
24565: - Code Editor: Compile with <F9> but also Alt C in case <F9> isn't available;
24566: - Code: if you cant run a function try the second one, for ex. Voice() -Voice2(),inc() - incl()
24567: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
24568: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
24569: - IDE menu /Help/Tools/ open the Task Manager
24570:
24571: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
24572: - Add on when no browser is available start /Options/Add ons/Easy Browser
24573: - Add on SOAP Tester with SOP POST File
24574: - Add on IP Protocol Sniffer with List View
24575: - Add on OpenGL mX Robot Demo for android
24576: - Add on Checkers Game, Add on Oscilloscope /View GEO Map View3
24577:
24578: - Menu: Help/Tools as a Tool Section with DOS Opener
24579: - Menu Editor: export the code as RTF File
24580: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
24581: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
24582: - Context: Auto Detect of Syntax depending on file extension
24583: - Code: some Windows API function start with w in the name like wGetAtomName();
24584: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
24585: - IDE File Check with menu ..View/File Changes/...
24586: - Context: Create a Header with Create Header in Navigator List at right window
24587: - Code: use SysErrorMessage to get a real Error Description, Ex.
24588: RemoveDir('c:\NoSuchFolder');writeln('System Error Message:' + SysErrorMessage(GetLastError));
24589: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
24590: - Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click on link
24591: - Editor: with <Ctrl+Alt+R> you can write in RTF format with RichEdit link
24592: - Menu: Check Help/Tools! you can use richedit, DOS Shell or Explorer
24593: - Menu: Start View/MyScript of ini file maxboxdef.ini [MYSOFT]= path of script
24594: - using DLL example in maXbox: //function: {*****}
24595: Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
24596:     cb: DWORD): BOOL; //stdcall;
24597:     External 'GetProcessMemoryInfo@psapi.dll stdcall';
24598: Function OpenProcess(dwDesiredAccess:DWORD;bInheritHandle:BOOL;dwProcessId: DWORD):THandle;
24599:     External 'OpenProcess@kernel32.dll stdcall';
24600:
24601: GCC Compile Ex Script
24602: procedure TFormMain_btnCompileClick(Sender: TObject);
24603: begin
24604:     AProcess:= TProcess.Create(nil);
24605:     try
24606:         AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"';
24607:         + ' -o "' + OpenDialog2.FileName + '"';
24608:         AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
24609:         AProcess.Execute;
24610:         Memo2.Lines.BeginUpdate;
24611:         Memo2.Lines.Clear;
24612:         Memo2.Lines.LoadFromStream(AProcess.Output);
24613:         Memo2.Lines.EndUpdate;
24614:     finally
24615:         AProcess.Free;
24616:     end;
24617:
24618: ref Stopwatch pattern snip
24619: Time1:= Time;
24620: writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
24621: if initAndStartBoard then
24622:     writeln('Filesize: '+inttostr(filesize(FILESAVE)));
24623:     writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
24624:     PrintF('%d %s',[Trunc((Time-Time1)*24),FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
24625:
24626: POST git-receive-pack (chunked)
24627: Pushing to https://github.com/maxkleiner/maxbox3.git
24628: To https://github.com/maxkleiner/maxbox3.git f127d21..c6a98da masterbox2 -> masterbox2
24629: updating local tracking ref 'refs/remotes/maxbox3Remote/masterbox2'
24630:
24631: -----

```

```

24632: History Shell Hell - Walk the Talk - CoolCode
24633: PCT Precompile Technology , mx4 ScriptStudio
24634: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
24635: DMath, devC, Graphics32, ExtPascal, mx4, LCL, CLX, FCL, CPort and more
24636: emax layers: system-package-component-unit-class-function-block
24637: new keywords def ref using maxCalcF UML: use case act class state seq pac comp dep - lib lab
24638: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
24639: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
24640: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
24641: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
24642: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
24643: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
24644: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
24645: SendBuffer, Color+Caption hack, ComboSet, SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
24646: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
24647: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
24648: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
24649: Inno Install and Setup Routines Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
24650: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
24651: Vfw (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
24652: 9 Color LED, LED Resources, Runtime LED, it + sr var, morse generator
24653: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
24654: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
24655: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
24656: ByteCode2, IPUtils2, GEOCode, CGI-Powtils, GPS_2, External App, Unit Converter
24657: Add 16 Units, 1 Slide, Tutor, Big Numbers (Decimals, TInteger), ModBusTCP, TGEOInfo
24658: Add 36 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, ACM, CDS, XMLDoc, DDE
24659: Add 64 Units, 12 Tutors, ChangeTracker, CPP+, OLEUtils2, xmldom, Chess4, 3DFrame, XMLRpc
24660: Add 77 Units, 12 Tutors, ChangeTracker, CPP+, OLEUtils2, xmldom, Chess4, 3DFrame, XMLRPC, X509
24661: Add 9 Units, 1 Tutor, InetUtils, Chron, REXX funcs
24662: Add 15 Units, 1 Tutor, Pipe Libraray2, KLog, FPplot42, KDialogs, NumEdit, KControls, Kronos
24663: add 12 units and 125 functions - Class Helper - KMemo RTF
24664: add 16 units and 365 functions- WMI Script Type Library - webbox
24665:
24666: Ref:
24667: https://unibe-ch.academia.edu/MaxKleiner
24668: http://www.slideshare.net/maxkleiner1
24669: http://www.scribd.com/max_kleiner
24670: http://www.delphiforfun.org/Programs/Utilities/index.htm
24671: http://www.slideshare.net/maxkleiner1
24672: http://s3.amazonaws.com/PreviewLinks/22959.html
24673: http://www.softwareschule.ch/arduino_training.pdf
24674: http://www.jrsoftware.org/isinfo.php
24675: http://www.be-precision.com/products/precision-builder/express/
24676: http://www.blaisepascal.eu/
24677: http://www.delhibasics.co.uk/
24678: http://www.youtube.com/watch?v=av89HAbqAsI
24679: http://www.angelfire.com/h15/delphizeus/modal.html
24680: http://www.retroarchive.org/garbo/pc/turbopas/index.html
24681: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
24682: http://delphi.org/2014/01/every-android-api-for-delphi/
24683: https://en.wikipedia.org/wiki/User:Maxkleiner
24684: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
24685: https://bitbucket.org/max_kleiner/maxbox3
24686: https://bitbucket.org/max_kleiner/maxbox3/downloads
24687: https://bitbucket.org/max_kleiner/maxbox3/wiki/maxBox%20Tutorials
24688: http://www.slideshare.net/maxkleiner1/codereview-topics
24689: http://www.abaecker.biz/images/abakus/images/TAbCompass.gif
24690: http://www.softpedia.com/get/Programming/Other-Programming-Files/maxbox.shtml
24691: http://max.kleiner.com/boxart.htm
24692:
24693: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chl=%s';
24694: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
 =renderBasicSearchNarrative&q=%s';
24695: UrlMapQuestAPIReverse='http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
 24696: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
24697:
24698: function OpenMap(const Data: string): boolean;
24699: var encURL: string;
24700: begin
24701:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPENcode(Data)]);
24702:   try //HttpGet(EncodedURL, mapStream); //WinInet
24703:     Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
24704:     //OpenDoc(Exepath+'openmapx.html');
24705:     S_ShellExecute(Exepath+'openmapx.html','','seCmdOpen');
24706:   finally encURL:= '';
24707:   end;
24708: end;
24709:
24710: procedure GetGEOMap(C_form,apath: string; const Data: string);
24711: var encodedURL: string; mapStream: TMemoryStream;
24712: begin //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPENcode(Data)]);
24713:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPENcode(Data)]);
24714:   mapStream:= TMemoryStream.create;
24715:   try
24716:     Wininet_HttpGet(EncodedURL, mapStream); //WinInet
24717:     mapStream.Position:= 0;
24718:     mapStream.Savetofile(apath); // OpenDoc(apath);
24719:     S_ShellExecute(apath,'',seCmdOpen);

```

```

24720:   finally
24721:     mapStream.Free;
24722:   end;
24723: end;
24724:
24725: Procedure BtnFactory(a,b,c,d:smallint; title,apic:string;
24726:                               var abtn:TBitBtn; anEvent:TNotifyEvent; afrm:TForm);
24727: begin
24728:   abtn:= TBitBtn.create(afrm);
24729:   with abtn do begin
24730:     parent:= afrm;
24731:     setBounds(a,b,c,d)
24732:     font.size:= 12;
24733:     glyph.LoadFromResourceName(HINSTANCE, apic);
24734:     mXButton(5,5,width, height,12,12,handle);
24735:     caption:= title;
24736:     onClick:= anEvent As TNotifyEvent;
24737:   end;
24738: end;
24739:
24740: function GetSystemDirectory2(var S: String): Boolean;
24741: var Len: Integer;
24742: begin
24743:   Len:= {Windows.}GetSystemDirectory('Nil', 0);
24744:   if Len > 0 then begin
24745:     SetLength(S, Len); //writeln(itoa(len))
24746:     Len:= {Windows.}GetSystemDirectory(PChar(S), Len);
24747:     SetLength(S, Len); //writeln(itoa(len))
24748:     Result:= Len > 0;
24749:   end else
24750:     Result:= False;
24751: end;
24752:
24753: Function RecordsetToXML2(const Recordset: variant): string;
24754: var
24755:   RS: Variant;
24756:   Stream: TStringStream;
24757: begin
24758:   Result := '';
24759:   //if Recordset = unassigned then Exit;
24760:   Stream:= TStringStream.Create('');
24761:   try
24762:     RS:= CreateOleObject('ADODB.Recordset');
24763:     RS:= Recordset;
24764:     RS.Save(TStreamAdapter.Create(stream,soReference) as IUnknown, adPersistXML);
24765:     Stream.Position := 0;
24766:     Result:= Stream.DataString;
24767:   finally
24768:     Stream.Free;
24769:   end;
24770: end;
24771:
24772: OSLibrary+
24773: function MySoundcard: Longint; external 'waveOutGetNumDevs@winmm.dll stdcall';
24774: function isSound: boolean; begin result:= mySoundcard > 0 end;
24775: function StringtoHex(Data: string): string;
24776: function GetAnsistringRefCount(const S: string): Cardinal;
24777: function WideStringToString(const ws: WideString; codePage: Word): AnsiString;
24778: function StringToWideString(const s: AnsiString; codePage: Word): WideString;
24779: procedure FreeObjectList(List: TObjectList);
24780: function SecondToTime(const Seconds: Cardinal): Double;
24781: function CopyDir2(const fromDir, toDir: string): Boolean;';
24782: function MoveDir(const fromDir, toDir: string): Boolean;';
24783: function DelDir(dir: string): Boolean;';
24784: procedure DeleteScansRect(Src, Dest: TBitmap; rs, rd: TRect);
24785: procedure FadeIn(ImageFileName: TFileName; aForm1: TForm);
24786: procedure FadeOut(ImageFileName: TFileName);
24787: procedure FadeOut32(const Bmp: TImage; Pause: Integer);
24788: function CheckBDEInstalled: Boolean; //IsBDE
24789: Procedure SaveStringToStream( const Str : String; Stream : TStream );
24790: Function LoadStringFromStream( Stream : TStream ) : String;
24791: Function WaitForSyncObject(SyncObject:THandle;Timeout:Cardinal;BlockInput:Boolean):Cardinal;
24792: Function ProcessMessage : Boolean';
24793: Procedure ProcessMessages( Timeout : DWORD ); //without application!
24794: Function GetNumberOfEventLogRecords(hEventLog:THandle; var NumberOfRecords: DWORD): BOOL;
24795: Function GetOldestEventLogRecord( hEventLog: THandle; var OldestRecord : DWORD ) : BOOL';
24796: ex.:EventLog:= RegisterEventSource('0','(maxbox3.exe')'; if GetNumberOfEventLogRecords(eventlog,recs)
24797: Procedure InitializeLocaleSupport; Procedure TerminateLocaleSupport';
24798: Function StrReplaceRegEx( const Subject, Pattern : AnsiString; Args : array of const : AnsiString );
24799: Function TickCountToDateTIme( Ticks : Cardinal ) : TDateTIme';
24800: Procedure OutputDebugStringmax( const S : String );
24801: Procedure OutputDebugFormat( const FmtStr : String; Args : array of const );
24802: Function IsAppRunningInDelphi : Boolean';
24803: Function RunExecutable( const AFileName : string; AWaitForIt : Boolean ) : DWORD';
24804: Function SystemCodePage : Integer';
24805: function CPUSpd: String;
24806: function CPUSpeed: String;
24807: function BigFib(n: integer): string; //BigFibo
24808: function BigFac(n: integer): string; //BigFact

```

```

24809: function UpTime: string;
24810: Function NetLogon(const Server,User,Password:WideString; out ErrorMessage:string) : Boolean';
24811: Function NetLogoff( const Server, User, Password : WideString) : Boolean';
24812: Procedure ErrorNamedPipe( const Message : string)';
24813: procedure BroadcastChange; //method that broadcasts the necessary message WM_SETTINGCHANGE
24814: Function WriteFile2( hFile : THandle; const Buffer, nNumberOfBytesToWrite : DWORD; var lpNumberOfBytesWritten : DWORD; lpOverlapped : Tobject) : BOOL';
24815: Function Readfile2( hFile : THandle; var Buffer, nNumberOfBytesToRead : DWORD; var lpNumberOfBytesRead : DWORD; lpOverlapped : Tobject) : BOOL';
24816: // Security helper functions
24817: procedure InitializeSecurity(var SA: TSecurityAttributes);
24818: procedure FinalizeSecurity(var SA: TSecurityAttributes);
24819: // Pipe helper functions
24820: procedure CheckPipeName(Value: string);
24821: procedure ClearOverlapped(var Overlapped: TOverlapped; ClearEvent: Boolean);
24822: procedure CloseHandleClear(var Handle: THandle);
24823: function ComputerName2: string;
24824: procedure DisconnectAndClose(Pipe: HPIPE; IsServer: Boolean = True);
24825: function EnumConsoleWindows(Window: HWND; lParam: Integer): BOOL; stdcall;
24826: procedure FlushMessages;
24827: function IsHandle(Handle: THandle): Boolean;
24828: procedure RaiseWindowsError;
24829: function UpTime: string; //OS uptime
24830: function CharSetToCP(ACharSet: TFontCharSet): Integer;
24831: function CPToCharSet(ACodePage: Integer): TFontCharSet;
24832: function TwipsToPoints(AValue: Integer): Integer;
24833: function PointsToTwips(AValue: Integer): Integer;
24834: procedure LoadGraphicFromResource(Graphic:TGraphic; const ResName:string; ResType:PChar);
24835: function SplitStr(sInput:string; Delimiter:string): TStringArray;
24836: function GetDataFromFile2(sFileName: AnsiString): AnsiString;
24837: function ExtractFileNameWithoutExt(const FileName: string): string;
24838: function SubstringCount (Substring : string; Str : string): Integer;
24839: function loadForm(vx, vy: smallint): TForm; //alias getForm()
24840: function getForm(vx, vy: smallint): TForm; //alias getForm()
24841: function getForm2(vx, vy: smallint; acolor: TColor; aName: string): TForm; //sizeable!
24842:
24843: procedure paintProcessingstar2(ppform: TForm);
24844: function ContentTypeGetExtn(Const Content: String; var CLSID: String): string;
24845: function ContentTypeFromExtn(Const Extension: String): string;
24846: function DateDiff2(Start, Stop : TDateTime) : int64;
24847: function checkSystem: string;';
24848: function getSystemReport: string;';
24849: function SystemCheck: string;';
24850:
24851:
24852: /////////////////////////////////
24853: // Object instance functions
24854: ///////////////////////////////
24855: function AllocateHWnd(Method: TWndMethod): HWND;
24856: procedure DeAllocateHWnd(Wnd: HWND);
24857: Function DeleteDC( DC : HDC ) : BOOL';
24858: Function DeleteMetaFile( p1 : HMETAFILE ) : BOOL';
24859: Function DeleteObject( p1 : HGDIOBJ ) : BOOL';
24860: function FormInstanceCount2(AFormClass: TFormClass): Integer;';
24861: function FormInstanceCount(const AFormClassName: string): Integer;';
24862: CL.AddClassN(CL.FindClass('Class of TForm'), 'TFormClass');
24863: Function ObjToStr( const O : TObject ) : String
24864:
24865:
24866: MessageBox(0, PChar('CPU speed is '+CPUSpd+' MHz'), 'CPU Speed Check', MB_ICONInformation+MB_OK);
24867: TKLogEvent', 'Procedure (Sender : TObject; Code : TKLogType; const Text : string)';
24868: AddRegisteredVariable('FormatSettings','TFormatSettings'); //at sysutils!
24869: AddRegisteredVariable('mouse','TMouse'); //at controls
24870: HPIPE', 'THandle');
24871:
24872: function ListIdentical2(l1,l2:TStringList): Boolean;
24873: begin Result:= False;
24874: if l1.count = l2.count then begin
24875: for it:= 0 to l1.count-1 do
24876: if (l1[it] <> l2[it]) then Exit;
24877: Result:= True;
24878: end;
24879: end;
24880:
24881: // Converts String To Hexadecimal Maybe usefull for a hex-editor
24882: // For example: Input = 'ABCD' Output = '41 42 43 44'
24883: function StringtoHex(Data: string): string;
24884: var i, i2: Integer; s: string;
24885: begin
24886: i2 := 1;
24887: for i := 1 to Length(Data) do begin
24888: Inc(i2);
24889: if i2 = 2 then
24890: s := s + ' '; i2 := 1;
24891: s := s + IntToHex(Ord(Data[i]), 2);
24892: end;
24893: Result := s;
24894: end;
24895:

```

```

24896: function XRTLIIsInMainThread: Boolean;
24897: begin
24898:   Result:= GetCurrentThreadID = MainThreadID;
24899: end;
24900:
24901:   computers := TStringList.Create;
24902:   FindComputers(computers)
24903:   for it:= 0 to computers.count-1 do
24904:     writeln(computers.strings[it]); computers.free;
24905:
24906:   s:= (Sender AS TStringGrid).Cells[col,row];
24907:   Sender AS TStringGrid).Canvas.Font.Color := clBlack;
24908:   twidth:= (Sender AS TStringGrid).Canvas.TextWidth('X');
24909:   (Sender AS TStringGrid).Canvas.TextRect(Rect, Rect.Left + twidth div 2,
24910:                                         YCenter(Rect, (Sender AS TStringGrid).Canvas, s), s)
24911:
24912: procedure | EExceptionsOnmaxXbox;
24913: var filename,emsg:string;
24914: begin
24915:   filename:='';
24916:   try
24917:     if filename = '' then
24918:       RaiseException(erCustomError, 'Exception: File name cannot be blank');
24919:     except
24920:       emsg:= ExceptionToString(ExceptionType, ExceptionParam);
24921:       //do act with the exception message i.e. email it or
24922:       //save to a log etc
24923:       writeln(emsg)
24924:     end;
24925:   end;
24926:
24927: procedure CallTicker(msecs: integer);
24928: begin
24929:   with TStopWatch.create do begin
24930:     start
24931:     | //Delay2StopWatch(msecs)
24932:     stop
24933:     writeln('Stop Watch CPU Tester: '+getValueStr)
24934:   end;
24935: end;
24936:
24937: function IsASCIIDigit(const Ch: Char): Boolean;
24938: begin
24939:   //Result := Ord(Ch) in [Ord('0')..Ord('9')];
24940:   Result:= (Ord(Ch) >= Ord('0')) And (ord(ch) <= Ord('9'));
24941: end;
24942:
24943: function RemainingBatteryPercent: Integer;
24944: var Stat: {Windows.TSystemPowerStatus;
24945: begin
24946:   {Windows.}GetSystemPowerStatus(Stat);
24947:   Result:= Stat.BatteryLifePercent;
24948:   if (Result < 0) Or (Result > 100) then
24949:     Result:= -1;
24950: end;
24951:
24952: // ****
24953: // TIdBaseComponent is the base class for all Indy components.
24954: // ****
24955: type
24956:   TIdBaseComponent = class(TComponent)
24957:   public
24958:     function GetVersion: string;
24959:     property Version: string read GetVersion;
24960:   published
24961:   end;
24962:
24963: if IsInternet then
24964:   wGetX('http://www.softwareschule.ch/download/maxbox_starter4.pdf', 'mywgettestpdf.pdf');
24965:
24966:   RegEx tester sr := '123.456.789-00$'
24967:   writeln(ReplaceRegExpr('\D',sr,'',true))
24968:   >>> 12345678900
24969:   //\D is a non-digit, and \W is a non-word character, both should work)
24970:   ref: lambda: a+b --> a.+ (b) in preparations
24971:   namedpipe1:= namedpipe as TNamedpipe;
24972:   printf('this is %.18f ', [maxCalc('ln(2)^e')]); this is 0.369248502937272178
24973:   printf('this is Area of r=1 %.18f ', [maxCalc('PI*(1^2)')]);
24974:   printf('this is Area of d=2r %.18f ', [maxCalc('PI/4*(2^2)')]);
24975:
24976: begin
24977:   initCRCarray2;
24978:   aStringStream := TStringStream.Create(filetoString(exepath+testFile));
24979:   try
24980:     //aStringStream.loadfromfile
24981:     writeln('CRC32_ : '+IntToHex(CalculateCRC32X(aStringStream), 8));
24982:   finally
24983:     aStringStream.Free;
24984:   end;

```

```

24985:     end;
24986:
24987: procedure TForm1Findwindow_Remote_NotePad;
24988: var
24989:   wnd: HWND;
24990:   i: Integer;
24991:   s: string;
24992: begin
24993:   wnd := FindWindow('notePad', '');
24994:   if wnd <> 0 then begin
24995:     wnd := FindWindowEx(wnd, 0, 'Edit', '');
24996:     // Write Text in NotePad.
24997:     s := 'Hello maXPad 3 ecotronics' + #13#10 + 'second line:' ;
24998:     for i := 1 to Length(s) do
24999:       SendMessage(wnd, WM_CHAR, Ord(s[i]), 0);
25000:     // Simulate Return Key.
25001:     PostMessage(wnd, WM_KEYDOWN, VK_RETURN, 0);
25002:     // Simulate Space.
25003:     PostMessage(wnd, WM_KEYDOWN, VK_SPACE, 0);
25004:     SendMessage(wnd, WM_CHAR, 5, 0);
25005:     //send time of F5 command in notePad!
25006:     PostMessage(wnd, WM_KEYDOWN, VK_F5, 0);
25007:   end;
25008: end;
25009:
25010: writeln(GetISODateString(1995,12,24))
25011: writeln(botostr.IsValidISODateTimeString(GetISODateString(1995,12,12), true)))
25012: writeln(MicrosoftCodePageToMIMECharset(1252))
25013: writeln(GetISOTimeString(12,45,33, true));
25014: writeln(botostr.IsValidISOTimeString(GetISOTimeString(12,45,33, true), true)))
25015:
25016:
25017:
25018: ****
25019: unit List asm internal end
25020: ****
25021: 01 unit RIRegister_Utils_Routines(exec); //Delphi
25022: 02 unit SIRegister_IdStrings; //Indy Sockets
25023: 03 unit RIRegister_niSTRING_Routines(Exec); //from RegEx
25024: 04 unit uPSI_fMain_Functions; //maXbox Open Tools API
25025: 05 unit IFSI_WinFormlpuzzle; //maXbox
25026: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
25027: 07 unit RegisterDateTimelibrary_R(exec); //Delphi
25028: 08 unit RIRegister_MathMax_Routines(exec); //Jedi & Delphi
25029: 09 unit RIRegister_IdGlobal_Routines(exec); //Indy Sockets
25030: 10 unit RIRegister_SysUtils_Routines(Exec); //Delphi
25031: 11 unit uPSI_IdTCPConnection; //Indy some functions
25032: 12 unit uPSCompiler.pas; //PS kernel functions
25033: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
25034: 14 unit uPSI_Printers.pas; //Delphi VCL
25035: 15 unit uPSI_MPlayer.pas; //Delphi VCL
25036: 16 unit uPSC_comobj; //COM Functions
25037: 17 unit uPSI_Clipbrd; //Delphi VCL
25038: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
25039: 19 unit uPSI_SqlExpr; //DBX3
25040: 20 unit uPSI_ADOdb; //ADODB
25041: 21 unit uPSI_StrHlpr; //String Helper Routines
25042: 22 unit uPSI_DateUtils; //Expansion to DateTimelib
25043: 23 unit uPSI_FileUtils; //Expansion to Sys/File Utils
25044: 24 unit JUutils / gsUtils; //Jedi / Metabase
25045: 25 unit JvFunctions_max; //Jedi Functions
25046: 26 unit HTTPParser; //Delphi VCL
25047: 27 unit HTTPUtil; //Delphi VCL
25048: 28 unit uPSI_XMLUtil; //Delphi VCL
25049: 29 unit uPSI_SOAPHTTPClient; //Delphi VCL SOAP WebService V3.5
25050: 30 unit uPSI_Contrns; //Delphi RTL Container of Classes
25051: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
25052: 32 unit uPSI_MyBigInt; //big integer class with Math
25053: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
25054: 34 unit Types_Variants; //Delphi Win32\rtl\sys
25055: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
25056: 36 unit uPSI_IdHashMessageDigest; //Indy Crypto;
25057: 37 unit uPSI_IdASN1Util; //Indy ASN1Utility Routines;
25058: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
25059: 39 unit uPSI_IdICmpClient; //Indy Ping ICMP
25060: 40 unit uPSI_IdHashMessageDigest_max //Indy Crypto &OpenSSL;
25061: 41 unit uPSI_FileCtrl; //Delphi RTL
25062: 42 unit uPSI_Outline; //Delphi VCL
25063: 43 unit uPSI_ScktComp; //Delphi RTL
25064: 44 unit uPSI_Calendar; //Delphi VCL
25065: 45 unit uPSI_VListView; //VListView;
25066: 46 unit uPSI_DBGrids; //Delphi VCL
25067: 47 unit uPSI_DBCtrls; //Delphi VCL
25068: 48 unit ide_debugoutput; //maXbox
25069: 49 unit uPSI_ComCtrls; //Delphi VCL
25070: 50 unit uPSC_stdCtrls+; //Delphi VCL
25071: 51 unit uPSI_Dialogs; //Delphi VCL
25072: 52 unit uPSI_StdConvs; //Delphi RTL
25073: 53 unit uPSI_DBClient; //Delphi RTL

```

```

25074: 54 unit uPSI_DBPlatform;                                //Delphi RTL
25075: 55 unit uPSI_Provider;                               //Delphi RTL
25076: 56 unit uPSI_FMTBcd;                                //Delphi RTL
25077: 57 unit uPSI_DBCGrids;                             //Delphi VCL
25078: 58 unit uPSI_CDSUtil;                              //MIDAS
25079: 59 unit uPSI_VarHlpr;                            //Delphi RTL
25080: 60 unit uPSI_ExtDlgs;                           //Delphi VCL
25081: 61 unit sdpStopwatch;                         //maXbox
25082: 62 unit uPSI_JclStatistics;                      //JCL
25083: 63 unit uPSI_JclLogic;                           //JCL
25084: 64 unit uPSI_JclMiscel;                          //JCL
25085: 65 unit uPSI_JclMath_max;                        //JCL RTL
25086: 66 unit uPSI_uTPLb_StreamUtils;                  //LockBox 3
25087: 67 unit uPSI_MathUtils;                          //BCB
25088: 68 unit uPSI_JclMultimedia;                     //JCL
25089: 69 unit uPSI_WideStrUtils;                      //Delphi API/RTL
25090: 70 unit uPSI_GraphUtil;                           //Delphi RTL
25091: 71 unit uPSI_TypeTrans;                          //Delphi RTL
25092: 72 unit uPSI_HTTPApp;                           //Delphi VCL
25093: 73 unit uPSI_DBWeb;                            //Delphi VCL
25094: 74 unit uPSI_DBBdeWeb;                          //Delphi VCL
25095: 75 unit uPSI_DBXpressWeb;                       //Delphi VCL
25096: 76 unit uPSI_ShadowWnd;                         //Delphi VCL
25097: 77 unit uPSI_ToolWin;                           //Delphi VCL
25098: 78 unit uPSI_Tabs;                             //Delphi VCL
25099: 79 unit uPSI_JclGraphUtils;                     //JCL
25100: 80 unit uPSI_JclCounter;                         //JCL
25101: 81 unit uPSI_JclSysInfo;                        //JCL
25102: 82 unit uPSI_JclSecurity;                       //JCL
25103: 83 unit uPSI_JclFileUtils;                      //JCL
25104: 84 unit uPSI_IdUserAccounts;                   //Indy
25105: 85 unit uPSI_IdAuthentication;                 //Indy
25106: 86 unit uPSI_uTPLb_AES;                         //LockBox 3
25107: 87 unit uPSI_IdHashSHA1;                        //LockBox 3
25108: 88 unit uPSI_BlockCipher;                      //LockBox 3
25109: 89 unit uPSI_ValEdit.pas;                      //Delphi VCL
25110: 90 unit uPSI_JvVCLUtils;                        //JCL
25111: 91 unit uPSI_JvDBUtil;                          //JCL
25112: 92 unit uPSI_JvDBUtils;                         //JCL
25113: 93 unit uPSI_JvAppUtils;                        //JCL
25114: 94 unit uPSI_JvCtrlUtils;                      //JCL
25115: 95 unit uPSI_JvFormToHtml;                     //JCL
25116: 96 unit uPSI_JvParsing;                         //JCL
25117: 97 unit uPSI_SerDlgs;                           //Toolbox
25118: 98 unit uPSI_Serial;                           //Toolbox
25119: 99 unit uPSI_JvComponent;                     //JCL
25120: 100 unit uPSI_JvCalc;                           //JCL
25121: 101 unit uPSI_JvBdeUtils;                      //JCL
25122: 102 unit uPSI_JvDateUtil;                      //JCL
25123: 103 unit uPSI_JvGenetic;                       //JCL
25124: 104 unit uPSI_JclBase;                          //JCL
25125: 105 unit uPSI_JvUtils;                          //JCL
25126: 106 unit uPSI_JvStringUtil;                    //JCL
25127: 107 unit uPSI_JvStringUtil;                    //JCL
25128: 108 unit uPSI_JvFileUtil;                      //JCL
25129: 109 unit uPSI_JvMemoryInfos;                  //JCL
25130: 110 unit uPSI_JvComputerInfo;                 //JCL
25131: 111 unit uPSI_JvgCommClasses;                //JCL
25132: 112 unit uPSI_JvgLogics;                      //JCL
25133: 113 unit uPSI_JvLED;                           //JCL
25134: 114 unit uPSI_JvTurtle;                        //JCL
25135: 115 unit uPSI_SortThds; unit uPSI_ThSort;     //maXbox
25136: 116 unit uPSI_JvgUtils;                        //JCL
25137: 117 unit uPSI_JvExprParser;                   //JCL
25138: 118 unit uPSI_HexDump;                         //Borland
25139: 119 unit uPSI_DBLogDlg;                       //VCL
25140: 120 unit uPSI_SqlTimSt;                        //RTL
25141: 121 unit uPSI_JvHtmlParser;                   //JCL
25142: 122 unit uPSI_JvgXMLSerializer;              //JCL
25143: 123 unit uPSI_JvJCLUtils;                     //JCL
25144: 124 unit uPSI_JvStrings;                      //JCL
25145: 125 unit uPSI_uTPLb_IntegerUtils;             //TurboPower
25146: 126 unit uPSI_uTPLb_HugeCardinal;            //TurboPower
25147: 127 unit uPSI_uTPLb_HugeCardinalUtils;       //TurboPower
25148: 128 unit uPSI_SynRegExpr;                    //SynEdit
25149: 129 unit uPSI_StUtils;                        //SysTools4
25150: 130 unit uPSI_StToHTML;                       //SysTools4
25151: 131 unit uPSI_StStrms;                        //SysTools4
25152: 132 unit uPSI_StFIN;                          //SysTools4
25153: 133 unit uPSI_StAstroP;                      //SysTools4
25154: 134 unit uPSI_StStat;                         //SysTools4
25155: 135 unit uPSI_StNetCon;                      //SysTools4
25156: 136 unit uPSI_StDecMth;                      //SysTools4
25157: 137 unit uPSI_StOStr;                         //SysTools4
25158: 138 unit uPSI_StPtrns;                        //SysTools4
25159: 139 unit uPSI_StNetMsg;                      //SysTools4
25160: 140 unit uPSI_StMath;                         //SysTools4
25161: 141 unit uPSI_StExpEng;                      //SysTools4
25162: 142 unit uPSI_StCRC;                          //SysTools4

```

```

25163: 143 unit uPSI_StExport; //SysTools4
25164: 144 unit uPSI_StExpLog; //SysTools4
25165: 145 unit uPSI_AcLnList; //Delphi VCL
25166: 146 unit uPSI_jpeg; //Borland
25167: 147 unit uPSI_StRandom; //SysTools4
25168: 148 unit uPSI_StDict; //SysTools4
25169: 149 unit uPSI_StBCD; //SysTools4
25170: 150 unit uPSI_StTxtDat; //SysTools4
25171: 151 unit uPSI_StRegEx; //SysTools4
25172: 152 unit uPSI_IMouse; //VCL
25173: 153 unit uPSI_SyncObjs; //VCL
25174: 154 unit uPSI_AsyncCalls; //Hausladen
25175: 155 unit uPSI_ParallelJobs; //Saraiva
25176: 156 unit uPSI_Variants; //VCL
25177: 157 unit uPSI_VarCmplx; //VCL Wolfram
25178: 158 unit uPSI_TDSSchema; //VCL
25179: 159 unit uPSI_ShLwApi; //Brakel
25180: 160 unit uPSI_IBUtils; //VCL
25181: 161 unit uPSI_CheckLst; //VCL
25182: 162 unit uPSI_JvSimpleXml; //JCL
25183: 163 unit uPSI_JclSimpleXml; //JCL
25184: 164 unit uPSI_JvXmlDatabase; //JCL
25185: 165 unit uPSI_JvMaxPixel; //JCL
25186: 166 unit uPSI_JvItemsSearchs; //JCL
25187: 167 unit uPSI_StExpEng2; //SysTools4
25188: 168 unit uPSI_StGenLog; //SysTools4
25189: 169 unit uPSI_JvLogFile; //Jcl
25190: 170 unit uPSI_CPort; //ComPort Lib v4.11
25191: 171 unit uPSI_CPortCtl; //ComPort
25192: 172 unit uPSI_CPortEsc; //ComPort
25193: 173 unit BarCodeScanner; //ComPort
25194: 174 unit uPSI_JvGraph; //JCL
25195: 175 unit uPSI_JvComCtrls; //JCL
25196: 176 unit uPSI_GUITesting; //D Unit
25197: 177 unit uPSI_JvFindFiles; //JCL
25198: 178 unit uPSI_StSystem; //SysTools4
25199: 179 unit uPSI_JvKeyboardStates; //JCL
25200: 180 unit uPSI_JvMail; //JCL
25201: 181 unit uPSI_JclConsole; //JCL
25202: 182 unit uPSI_JclLANman; //JCL
25203: 183 unit uPSI_IdCustomHTTPServer; //Indy
25204: 184 unit IdHTTPServer; //Indy
25205: 185 unit uPSI_IdTCPServer; //Indy
25206: 186 unit uPSI_IdSocketHandle; //Indy
25207: 187 unit uPSI_IdIOHandlerSocket; //Indy
25208: 188 unit IdIOHandler; //Indy
25209: 189 unit uPSI_cutils; //Bloodshed
25210: 190 unit uPSI_BoldUtils; //boldsoft
25211: 191 unit uPSI_IdSimpleServer; //Indy
25212: 192 unit uPSI_IdSSLOpenSSL; //Indy
25213: 193 unit uPSI_IdMultipartFormData; //Indy
25214: 194 unit uPSI_SynURIOpener; //SynEdit
25215: 195 unit uPSI_PerlRegEx; //PCRE
25216: 196 unit uPSI_IdHeaderList; //Indy
25217: 197 unit uPSI_StFirst; //SysTools4
25218: 198 unit uPSI_JvCtrls; //JCL
25219: 199 unit uPSI_IdTrivialFTPBase; //Indy
25220: 200 unit uPSI_IdTrivialFTP; //Indy
25221: 201 unit uPSI_IdUDPBase; //Indy
25222: 202 unit uPSI_IdUDPClient; //Indy
25223: 203 unit uPSI_utypes; //for DMath.DLL
25224: 204 unit uPSI_ShellAPI; //Borland
25225: 205 unit uPSI_IdRemoteCMDClient; //Indy
25226: 206 unit uPSI_IdRemoteCMDServer; //Indy
25227: 207 unit IdRexecServer; //Indy
25228: 208 unit IdRexec; (unit uPSI_IdRexec;) //Indy
25229: 209 unit IdUDPServer; //Indy
25230: 210 unit IdTimeUDPServer; //Indy
25231: 211 unit IdTimeServer; //Indy
25232: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
25233: 213 unit uPSI_IdIPWatch; //Indy
25234: 214 unit uPSI_IdIrcServer; //Indy
25235: 215 unit uPSI_IdMessageCollection; //Indy
25236: 216 unit uPSI_cPEM; //Fundamentals 4
25237: 217 unit uPSI_cFundamentUtils; //Fundamentals 4
25238: 218 unit uPSI_uwinplot; //DMath
25239: 219 unit uPSI_xrtl_util_CPUUtils; //ExtentedRTL
25240: 220 unit uPSI_GR32_System; //Graphics32
25241: 221 unit uPSI_cFileUtils; //Fundamentals 4
25242: 222 unit uPSI_cDateTime; (timemachine) //Fundamentals 4
25243: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
25244: 224 unit uPSI_cRandom; //Fundamentals 4
25245: 225 unit uPSI_ueval; //DMath
25246: 226 unit uPSI_xrtl_net_URIUtils; //ExtendedRTL
25247: 227 unit xrtl_net_URIUtils; //ExtendedRTL
25248: 228 unit uPSI_ufft; (FFT) //DMath
25249: 229 unit uPSI_DBXChannel; //Delphi
25250: 230 unit uPSI_DBXIndyChannel; //Delphi Indy
25251: 231 unit uPSI_xrtl_util_COMCat; //ExtendedRTL

```

```

25252: 232 unit uPSI_xrtl_util_StrUtils; //ExtendedRTL
25253: 233 unit uPSI_xrtl_util_VariantUtils; //ExtendedRTL
25254: 234 unit uPSI_xrtl_util_FileUtils; //ExtendedRTL
25255: 235 unit xrtl_util_Compat; //ExtendedRTL
25256: 236 unit uPSI_OleAuto; //Borland
25257: 237 unit uPSI_xrtl_util_COMUtils; //ExtendedRTL
25258: 238 unit uPSI_CmAdmCtl; //Borland
25259: 239 unit uPSI_ValEdit2; //VCL
25260: 240 unit uPSI_GR32; //Graphics32
25261: 241 unit uPSI_GR32_Image; //Graphics32
25262: 242 unit uPSI_xrtl_util_TimeUtils; //ExtendedRTL
25263: 243 unit uPSI_xrtl_util_TimeZone; //ExtendedRTL
25264: 244 unit uPSI_xrtl_util_TimeStamp; //ExtendedRTL
25265: 245 unit uPSI_xrtl_util_Map; //ExtendedRTL
25266: 246 unit uPSI_xrtl_util_Set; //ExtendedRTL
25267: 247 unit uPSI_CPortMonitor; //ComPort
25268: 248 unit uPSI_StIniStm; //SysTools4
25269: 249 unit uPSI_GR32_ExtImage; //Graphics32
25270: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32
25271: 251 unit uPSI_GR32_Rasterizers; //Graphics32
25272: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
25273: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
25274: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
25275: 255 unit uPSI_FlatSB; //VCL
25276: 256 unit uPSI_JvAnalogClock; //JCL
25277: 257 unit uPSI_JvAlarms; //JCL
25278: 258 unit uPSI_JvSQLS; //JCL
25279: 259 unit uPSI_JvDBSecur; //JCL
25280: 260 unit uPSI_JvDBQRE; //JCL
25281: 261 unit uPSI_JvStarfield; //JCL
25282: 262 unit uPSI_JVCLMiscal; //JCL
25283: 263 unit uPSI_JvProfiler32; //JCL
25284: 264 unit uPSI_JvDirectories; //JCL
25285: 265 unit uPSI_JclSchedule; //JCL
25286: 266 unit uPSI_JclSvcCtrl; //JCL
25287: 267 unit uPSI_JvSoundControl; //JCL
25288: 268 unit uPSI_JvBDESQLScript; //JCL
25289: 269 unit uPSI_JvgDigits; //JCL>
25290: 270 unit uPSI_ImgList; //TCustomImageList
25291: 271 unit uPSI_JclMIDI; //JCL>
25292: 272 unit uPSI_JclWinMidi; //JCL>
25293: 273 unit uPSI_JclNTFS; //JCL>
25294: 274 unit uPSI_JclAppInst; //JCL>
25295: 275 unit uPSI_JvRle; //JCL>
25296: 276 unit uPSI_JvRas32; //JCL>
25297: 277 unit uPSI_JvImageDrawThread; //JCL>
25298: 278 unit uPSI_JvImageWindow; //JCL>
25299: 279 unit uPSI_JvTransparentForm; //JCL>
25300: 280 unit uPSI_JvWinDialogs; //JCL>
25301: 281 unit uPSI_JvSimLogic; //JCL>
25302: 282 unit uPSI_JvSimIndicator; //JCL>
25303: 283 unit uPSI_JvSimPID; //JCL>
25304: 284 unit uPSI_JvSimPIDLinker; //JCL>
25305: 285 unit uPSI_IdRFCReply; //Indy
25306: 286 unit uPSI_IdIdent; //Indy
25307: 287 unit uPSI_IdIdentServer; //Indy
25308: 288 unit uPSI_JvPatchFile; //JCL
25309: 289 unit uPSI_StNetPfm; //SysTools4
25310: 290 unit uPSI_StNet; //SysTools4
25311: 291 unit uPSI_JclPeImage; //JCL
25312: 292 unit uPSI_JclPrint; //JCL
25313: 293 unit uPSI_JclMime; //JCL
25314: 294 unit uPSI_JvRichEdit; //JCL
25315: 295 unit uPSI_JvDBRichEd; //JCL
25316: 296 unit uPSI_JvDice; //JCL
25317: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
25318: 298 unit uPSI_JvDirFrm; //JCL
25319: 299 unit uPSI_JvDualList; //JCL
25320: 300 unit uPSI_JvSwitch; //JCL
25321: 301 unit uPSI_JvTimerLst; //JCL
25322: 302 unit uPSI_JvMemTable; //JCL
25323: 303 unit uPSI_JvObjStr; //JCL
25324: 304 unit uPSI_StLArr; //SysTools4
25325: 305 unit uPSI_StWmDCpy; //SysTools4
25326: 306 unit uPSI_StText; //SysTools4
25327: 307 unit uPSI_StNTLog; //SysTools4
25328: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
25329: 309 unit uPSI_JvImagPrvw; //JCL
25330: 310 unit uPSI_JvFormPatch; //JCL
25331: 311 unit uPSI_JvPicClip; //JCL
25332: 312 unit uPSI_JvDataConv; //JCL
25333: 313 unit uPSI_JvCpuUsage; //JCL
25334: 314 unit uPSI_JclUnitConv_mx2; //JCL
25335: 315 unit JvDualListForm; //JCL
25336: 316 unit uPSI_JvCpuUsage2; //JCL
25337: 317 unit uPSI_JvParserForm; //JCL
25338: 318 unit uPSI_JvJanTreeView; //JCL
25339: 319 unit uPSI_JvTransLED; //JCL
25340: 320 unit uPSI_JvPlaylist; //JCL

```

```

25341: 321 unit uPSI_JvFormAutoSize; //JCL
25342: 322 unit uPSI_JvYearGridEditForm; //JCL
25343: 323 unit uPSI_JvMarkupCommon; //JCL
25344: 324 unit uPSI_JvChart; //JCL
25345: 325 unit uPSI_JvXPCore; //JCL
25346: 326 unit uPSI_JvXPCoreUtils; //JCL
25347: 327 unit uPSI_StatsClasses; //mX4
25348: 328 unit uPSI_ExtCtrls2; //VCL
25349: 329 unit uPSI_JvUrlGrabbers; //JCL
25350: 330 unit uPSI_JvXmlTree; //JCL
25351: 331 unit uPSI_JvWavePlayer; //JCL
25352: 332 unit uPSI_JvUnicodeCanvas; //JCL
25353: 333 unit uPSI_JvTfuTools; //JCL
25354: 334 unit uPSI_IdServerIOHandler; //Indy
25355: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
25356: 336 unit uPSI_IdMessageCoder; //Indy
25357: 337 unit uPSI_IdMessageCoderMIME; //Indy
25358: 338 unit uPSI_IdMIMETypes; //Indy
25359: 339 unit uPSI_JvConverter; //JCL
25360: 340 unit uPSI_JvCsvParse; //JCL
25361: 341 unit uPSI_umat; unit uPSI_ugamma; //DMath
25362: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
25363: 343 unit uPSI_JvDBGridExport; //JCL
25364: 344 unit uPSI_JvgExport; //JCL
25365: 345 unit uPSI_JvSerialMaker; //JCL
25366: 346 unit uPSI_JvWin32; //JCL
25367: 347 unit uPSI_JvPaintFX; //JCL
25368: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
25369: 349 unit uPSI_JvValidators; (preview) //JCL
25370: 350 unit uPSI_JvNTEventLog; //JCL
25371: 351 unit uPSI_ShellZipTool; //mX4
25372: 352 unit uPSI_JvJoystick; //JCL
25373: 353 unit uPSI_JvMailSlots; //JCL
25374: 354 unit uPSI_JclComplex; //JCL
25375: 355 unit uPSI_SynPdf; //Synopse
25376: 356 unit uPSI_Registry; //VCL
25377: 357 unit uPSI_TlHelp32; //VCL
25378: 358 unit uPSI_JclRegistry; //JCL
25379: 359 unit uPSI_JvAirBrush; //JCL
25380: 360 unit uPSI_mORMotReport; //Synopse
25381: 361 unit uPSI_JclLocales; //JCL
25382: 362 unit uPSI_SynEdit; //SynEdit
25383: 363 unit uPSI_SynEditTypes; //SynEdit
25384: 364 unit uPSI_SynMacroRecorder; //SynEdit
25385: 365 unit uPSI_LongIntList; //SynEdit
25386: 366 unit uPSI_devutils; //DevC
25387: 367 unit uPSI_SynEditMiscClasses; //SynEdit
25388: 368 unit uPSI_SynEditRegexSearch; //SynEdit
25389: 369 unit uPSI_SynEditHighlighter; //SynEdit
25390: 370 unit uPSI_SynHighlighterPas; //SynEdit
25391: 371 unit uPSI_JvSearchFiles; //JCL
25392: 372 unit uPSI_SynHighlighterAny; //Lazarus
25393: 373 unit uPSI_SynEditKeyCmds; //SynEdit
25394: 374 unit uPSI_SynEditMiscProcs; //SynEdit
25395: 375 unit uPSI_SynEditKbdHandler; //SynEdit
25396: 376 unit uPSI_JvAppInst; //JCL
25397: 377 unit uPSI_JvAppEvent; //JCL
25398: 378 unit uPSI_JvAppCommand; //JCL
25399: 379 unit uPSI_JvAnimTitle; //JCL
25400: 380 unit uPSI_JvAnimatedImage; //JCL
25401: 381 unit uPSI_SynEditExport; //SynEdit
25402: 382 unit uPSI_SynExportHTML; //SynEdit
25403: 383 unit uPSI_SynExportRTF; //SynEdit
25404: 384 unit uPSI_SynEditSearch; //SynEdit
25405: 385 unit uPSI_fMain_back; //mXbox;
25406: 386 unit uPSI_JvZoom; //JCL
25407: 387 unit uPSI_FMrand; //PM
25408: 388 unit uPSI_JvSticker; //JCL
25409: 389 unit uPSI_XmlVerySimple; //mX4
25410: 390 unit uPSI_Services; //ExtPascal
25411: 391 unit uPSI_ExtPascalUtils; //ExtPascal
25412: 392 unit uPSI_SocketsDelphi; //ExtPascal
25413: 393 unit uPSI_StBarC; //SysTools
25414: 394 unit uPSI_StDbBarC; //SysTools
25415: 395 unit uPSI_StBarPN; //SysTools
25416: 396 unit uPSI_StDbPNBC; //SysTools
25417: 397 unit uPSI_StDb2DBC; //SysTools
25418: 398 unit uPSI_StMoney; //SysTools
25419: 399 unit uPSI_JvForth; //JCL
25420: 400 unit uPSI_RestRequest; //mX4
25421: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
25422: 402 unit uPSI_JvXmlDatabase; //update //JCL
25423: 403 unit uPSI_StAstro; //SysTools
25424: 404 unit uPSI_StSort; //SysTools
25425: 405 unit uPSI_StDate; //SysTools
25426: 406 unit uPSI_StDateSt; //SysTools
25427: 407 unit uPSI_StBase; //SysTools
25428: 408 unit uPSI_StVInfo; //SysTools
25429: 409 unit uPSI_JvBrowseFolder; //JCL

```

```

25430: 410 unit uPSI_JvBoxProcs; //JCL
25431: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
25432: 412 unit uPSI_usimann; (unit ugenalgr;) //DMath
25433: 413 unit uPSI_JvHighlighter; //JCL
25434: 414 unit uPSI_Diff; //mX4
25435: 415 unit uPSI_SpringWinAPI; //DSpring
25436: 416 unit uPSI_StBits; //SysTools
25437: 417 unit uPSI_TomDBQue; //mX4
25438: 418 unit uPSI_MultilangTranslator; //mX4
25439: 419 unit uPSI_HyperLabel; //mX4
25440: 420 unit uPSI_Starter; //mX4
25441: 421 unit uPSI_FileAssocs; //devC
25442: 422 unit uPSI_devFileMonitorX; //devC
25443: 423 unit uPSI_devrund; //devC
25444: 424 unit uPSI_devExec; //devC
25445: 425 unit uPSI_oxsUtils; //devC
25446: 426 unit uPSI_DosCommand; //devC
25447: 427 unit uPSI_CppTokenizer; //devC
25448: 428 unit uPSI_JvHLPParser; //devC
25449: 429 unit uPSI_JclMapi; //JCL
25450: 430 unit uPSI_JclShell; //JCL
25451: 431 unit uPSI_JclCOM; //JCL
25452: 432 unit uPSI_GR32_Math; //Graphics32
25453: 433 unit uPSI_GR32_LowLevel; //Graphics32
25454: 434 unit uPSI_SimpleHl; //mX4
25455: 435 unit uPSI_GR32_Filters; //Graphics32
25456: 436 unit uPSI_GR32_VectorMaps; //Graphics32
25457: 437 unit uPSI_cXMLFunctions; //Fundamentals 4
25458: 438 unit uPSI_JvTimer; //JCL
25459: 439 unit uPSI_cHTTPUtils; //Fundamentals 4
25460: 440 unit uPSI_cTLSUtils; //Fundamentals 4
25461: 441 unit uPSI_JclGraphics; //JCL
25462: 442 unit uPSI_JclSynch; //JCL
25463: 443 unit uPSI_IdTelnet; //Indy
25464: 444 unit uPSI_IdTelnetServer; //Indy
25465: 445 unit uPSI_IdEcho; //Indy
25466: 446 unit uPSI_IdEchoServer; //Indy
25467: 447 unit uPSI_IdEchoUDP; //Indy
25468: 448 unit uPSI_IdEchoUDPServer; //Indy
25469: 449 unit uPSI_IdSocks; //Indy
25470: 450 unit uPSI_IdAntiFreezeBase; //Indy
25471: 451 unit uPSI_IdHostnameServer; //Indy
25472: 452 unit uPSI_IdTunnelCommon; //Indy
25473: 453 unit uPSI_IdTunnelMaster; //Indy
25474: 454 unit uPSI_IdTunnelSlave; //Indy
25475: 455 unit uPSI_IdRSH; //Indy
25476: 456 unit uPSI_IdRSHServer; //Indy
25477: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi
25478: 458 unit uPSI_MapReader; //devC
25479: 459 unit uPSI_LibTar; //devC
25480: 460 unit uPSI_IdStack; //Indy
25481: 461 unit uPSI_IdBlockCipherIntercept; //Indy
25482: 462 unit uPSI_IdChargenServer; //Indy
25483: 463 unit uPSI_IdFTPServer; //Indy
25484: 464 unit uPSI_IdException; //Indy
25485: 465 unit uPSI_utexplot; //DMath
25486: 466 unit uPSI_uwinstr; //DMath
25487: 467 unit uPSI_VarRecUtils; //devC
25488: 468 unit uPSI_JvStringListToHtml; //JCL
25489: 469 unit uPSI_JvStringHolder; //JCL
25490: 470 unit uPSI_IdCoder; //Indy
25491: 471 unit uPSI_SynHighlighterDfm; //Synedit
25492: 472 unit uHighlighterProcs; in 471 //Synedit
25493: 473 unit uPSI_LazFileUtils; //LCL
25494: 474 unit uPSI_IDECmdLine; //LCL
25495: 475 unit uPSI_lazMasks; //LCL
25496: 476 unit uPSI_ip_misc; //mX4
25497: 477 unit uPSI_Barcodes; //LCL
25498: 478 unit uPSI_SimpleXML; //LCL
25499: 479 unit uPSI_JclIniFiles; //JCL
25500: 480 unit uPSI_D2XXUnit; {$_$X-} //FTDI
25501: 481 unit uPSI_JclDateTime; //JCL
25502: 482 unit uPSI_JclEDI; //JCL
25503: 483 unit uPSI_JclMiscel2; //JCL
25504: 484 unit uPSI_JclValidation; //JCL
25505: 485 unit uPSI_JclAnsiStrings; {-PString} //JCL
25506: 486 unit uPSI_SynEditMiscProcs2; //Synedit
25507: 487 unit uPSI_JclStreams; //JCL
25508: 488 unit uPSI_QRCode; //mX4
25509: 489 unit uPSI_BlockSocket; //ExtPascal
25510: 490 unit uPSI_Masks_Utils //VCL
25511: 491 unit uPSI_synautil; //Synapse!
25512: 492 unit uPSI_JclMath_Class; //JCL RTL
25513: 493 unit ugamdist; //Gamma function //DMath
25514: 494 unit uibeta, ucorrel; //IBeta //DMath
25515: 495 unit uPSI_SRMgr; //mX4
25516: 496 unit uPSI_HotLog; //mX4
25517: 497 unit uPSI_DebugBox; //mX4
25518: 498 unit uPSI_ustrings; //DMath

```

```

25519: 499 unit uPSI_uregtest; //DMath
25520: 500 unit uPSI_usimplex; //DMath
25521: 501 unit uPSI_uhyper; //Indy
25522: 502 unit uPSI_IdHL7; //Indy
25523: 503 unit uPSI_IdIPMCastBase, //Indy
25524: 504 unit uPSI_IdIPMCastServer; //Indy
25525: 505 unit uPSI_IdIPMCastClient; //Indy
25526: 506 unit uPSI_unlfit; //nlregression //DMath
25527: 507 unit uPSI_IdRawHeaders; //Indy
25528: 508 unit uPSI_IdRawClient; //Indy
25529: 509 unit uPSI_IdRawFunctions; //Indy
25530: 510 unit uPSI_IdTCPStream; //Indy
25531: 511 unit uPSI_IdSNPP; //Indy
25532: 512 unit uPSI_St2DBarC; //SysTools
25533: 513 unit uPSI_ImageWin; //FTL //VCL
25534: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
25535: 515 unit uPSI_GraphWin; //FTL //VCL
25536: 516 unit uPSI_actionMain; //FTL //VCL
25537: 517 unit uPSI_StSpawn; //SysTools
25538: 518 unit uPSI_CtlPanel; //VCL
25539: 519 unit uPSI_IdLPR; //Indy
25540: 520 unit uPSI_SockRequestInterpreter; //Indy
25541: 521 unit uPSI_ulambert; //DMath
25542: 522 unit uPSI_ucholesk; //DMath
25543: 523 unit uPSI_SimpleDS; //VCL
25544: 524 unit uPSI_DBXSqlScanner; //VCL
25545: 525 unit uPSI_DBXMetaDataTable; //VCL
25546: 526 unit uPSI_Chart; //TEE
25547: 527 unit uPSI_TeeProcs; //TEE
25548: 528 unit mXBDEUtils; //mX4
25549: 529 unit uPSI_MDIEdit; //VCL
25550: 530 unit uPSI_CopyPrsr; //VCL
25551: 531 unit uPSI_SockApp; //VCL
25552: 532 unit uPSI_AppEvnts; //VCL
25553: 533 unit uPSI_ExtActns; //VCL
25554: 534 unit uPSI_TeEngine; //TEE
25555: 535 unit uPSI_CoolMain; //browser //VCL
25556: 536 unit uPSI_StCRC; //SysTools
25557: 537 unit uPSI_StDecMth2; //SysTools
25558: 538 unit uPSI_frmExportMain; //Synedit
25559: 539 unit uPSI_SynEdit; //Synedit
25560: 540 unit uPSI_SynEditWildcardSearch; //Synedit
25561: 541 unit uPSI_BoldComUtils; //BOLD
25562: 542 unit uPSI_BoldIsoDateTime; //BOLD
25563: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
25564: 544 unit uPSI_BoldXMLRequests; //BOLD
25565: 545 unit uPSI_BoldStringList; //BOLD
25566: 546 unit uPSI_BoldFileHandler; //BOLD
25567: 547 unit uPSI_BoldContainers; //BOLD
25568: 548 unit uPSI_BoldQueryUserDlg; //BOLD
25569: 549 unit uPSI_BoldWinINet; //BOLD
25570: 550 unit uPSI_BoldQueue; //BOLD
25571: 551 unit uPSI_JvPcx; //JCL
25572: 552 unit uPSI_IdWhois; //Indy
25573: 553 unit uPSI_IdWhoIsServer; //Indy
25574: 554 unit uPSI_IdGopher; //Indy
25575: 555 unit uPSI_IdTimeStamp; //Indy
25576: 556 unit uPSI_IdDayTimeServer; //Indy
25577: 557 unit uPSI_IdDayTimeUDP; //Indy
25578: 558 unit uPSI_IdDayTimeUDPServer; //Indy
25579: 559 unit uPSI_IdDICTServer; //Indy
25580: 560 unit uPSI_IdDiscardServer; //Indy
25581: 561 unit uPSI_IdDiscardUDPServer; //Indy
25582: 562 unit uPSI_IdMappedFTP; //Indy
25583: 563 unit uPSI_IdMappedPortTCP; //Indy
25584: 564 unit uPSI_IdGopherServer; //Indy
25585: 565 unit uPSI_IdQotdServer; //Indy
25586: 566 unit uPSI_JvRgbToHtml; //JCL
25587: 567 unit uPSI_JvRemLog; //JCL
25588: 568 unit uPSI_JvSysComp; //JCL
25589: 569 unit uPSI_JvTMTL; //JCL
25590: 570 unit uPSI_JvWinampAPI; //JCL
25591: 571 unit uPSI_MSysUtils; //mX4
25592: 572 unit uPSI_ESBMaths; //ESB
25593: 573 unit uPSI_ESBMaths2; //ESB
25594: 574 unit uPSI_uLkJSON; //Lk
25595: 575 unit uPSI_ZURL; //Zeos
25596: 576 unit uPSI_ZSysUtils; //Zeos
25597: 577 unit unaUtils internals //UNA
25598: 578 unit uPSI_ZMatchPattern; //Zeos
25599: 579 unit uPSI_ZClasses; //Zeos
25600: 580 unit uPSI_ZCollections; //Zeos
25601: 581 unit uPSI_ZEncoding; //Zeos
25602: 582 unit uPSI_IdRawBase; //Indy
25603: 583 unit uPSI_IdNTLM; //Indy
25604: 584 unit uPSI_IdNNTP; //Indy
25605: 585 unit uPSI_usniffer; //PortScanForm //mX4
25606: 586 unit uPSI_IdCoderMIME; //Indy
25607: 587 unit uPSI_IdCoderUUE; //Indy

```

```

25608: 588 unit uPSI_IdCoderXXE;                                //Indy
25609: 589 unit uPSI_IdCoder3to4;                               //Indy
25610: 590 unit uPSI_IdCookie;                                 //Indy
25611: 591 unit uPSI_IdCookieManager;                            //Indy
25612: 592 unit uPSI_WDOSocketUtils;                            //WDOS
25613: 593 unit uPSI_WDOSPlcUtils;                             //WDOS
25614: 594 unit uPSI_WDOSPorts;                               //WDOS
25615: 595 unit uPSI_WDOSResolvers;                            //WDOS
25616: 596 unit uPSI_WDOSTimers;                              //WDOS
25617: 597 unit uPSI_WDOSPlcs;                               //WDOS
25618: 598 unit uPSI_WDOSPneumatics;                           //WDOS
25619: 599 unit uPSI_IdFingerServer;                           //Indy
25620: 600 unit uPSI_IdDNSResolver;                            //Indy
25621: 601 unit uPSI_IdHTTPWebBrokerBridge;                  //Indy
25622: 602 unit uPSI_IdIntercept;                             //Indy
25623: 603 unit uPSI_IdIPMCastBase;                            //Indy
25624: 604 unit uPSI_IdLogBase;                               //Indy
25625: 605 unit uPSI_IdIOHandlerStream;                         //Indy
25626: 606 unit uPSI_IdMappedPortUDP;                           //Indy
25627: 607 unit uPSI_IdQOTDUDPServer;                          //Indy
25628: 608 unit uPSI_IdQOTDUDP;                               //Indy
25629: 609 unit uPSI_IdSysLog;                                //Indy
25630: 610 unit uPSI_IdSysLogServer;                           //Indy
25631: 611 unit uPSI_IdSysLogMessage;                          //Indy
25632: 612 unit uPSI_IdTimeServer;                            //Indy
25633: 613 unit uPSI_IdTimeUDP;                               //Indy
25634: 614 unit uPSI_IdTimeUDPServer;                          //Indy
25635: 615 unit uPSI_IdUserAccounts;                           //Indy
25636: 616 unit uPSI_TextUtils;                               //mX4
25637: 617 unit uPSI_MandelbrotEngine;                          //mX4
25638: 618 unit uPSI_delphi_arduino_Unit1;                   //mX4
25639: 619 unit uPSI_TDSSchema2;                             //mX4
25640: 620 unit uPSI_fplotMain;                              //DMath
25641: 621 unit uPSI_FindFileIter;                            //mX4
25642: 622 unit uPSI_PppState;     (JclStrHashMap)           //PPP
25643: 623 unit uPSI_PppParser;                             //PPP
25644: 624 unit uPSI_PppLexer;                             //PPP
25645: 625 unit uPSI_PcharUtils;                            //PPP
25646: 626 unit uPSI_uJSON;                                //WU
25647: 627 unit uPSI_JclStrHashMap;                          //JCL
25648: 628 unit uPSI_JclHookExcept;                          //JCL
25649: 629 unit uPSI_EncdDecd;                             //VCL
25650: 630 unit uPSI_SockAppReg;                            //VCL
25651: 631 unit uPSI_PJHandle;                             //PJ
25652: 632 unit uPSI_PJEnvVars;                            //PJ
25653: 633 unit uPSI_PJPipe;                               //PJ
25654: 634 unit uPSI_PJPipeFilters;                          //PJ
25655: 635 unit uPSI_PJConsoleApp;                           //PJ
25656: 636 unit uPSI_UConsoleAppEx;                          //PJ
25657: 637 unit uPSI_DbSocketChannelNative;                 //VCL
25658: 638 unit uPSI_DbxDatagenerator;                     //VCL
25659: 639 unit uPSI_DBXClient;                            //VCL
25660: 640 unit uPSI_IdLogEvent;                            //Indy
25661: 641 unit uPSI_Reversi;                             //mX4
25662: 642 unit uPSI_Geometry;                            //mX4
25663: 643 unit uPSI_IdSMTPServer;                          //Indy
25664: 644 unit uPSI_Textures;                            //mX4
25665: 645 unit uPSI_IBX;                                 //VCL
25666: 646 unit uPSI_IWDBCommon;                           //VCL
25667: 647 unit uPSI_SortGrid;                            //mX4
25668: 648 unit uPSI_IB;                                 //VCL
25669: 649 unit uPSI_IBScript;                            //VCL
25670: 650 unit uPSI_JvCSVBaseControls;                  //JCL
25671: 651 unit uPSI_Jvg3DColors;                          //JCL
25672: 652 unit uPSI_JvHLEditor;   //beat                //JCL
25673: 653 unit uPSI_JvShellHook;                           //JCL
25674: 654 unit uPSI_DBCommon2;                           //VCL
25675: 655 unit uPSI_JvSHfileOperation;                  //JCL
25676: 656 unit uPSI_uFileexport;                           //mX4
25677: 657 unit uPSI_JvDialogs;                            //JCL
25678: 658 unit uPSI_JvDBTreeview;                          //JCL
25679: 659 unit uPSI_JvDBUltimgrid;                         //JCL
25680: 660 unit uPSI_JvDBQueryParamsForm;                 //JCL
25681: 661 unit uPSI_JvExControls;                          //JCL
25682: 662 unit uPSI_JvBDEMemTable;                         //JCL
25683: 663 unit uPSI_JvCommStatus;                          //JCL
25684: 664 unit uPSI_JvMailSlots2;                           //JCL
25685: 665 unit uPSI_JvgWinMask;                            //JCL
25686: 666 unit uPSI_StEclpse;                            //SysTools
25687: 667 unit uPSI_StMime;                             //SysTools
25688: 668 unit uPSI_StList;                             //SysTools
25689: 669 unit uPSI_StMerge;                            //SysTools
25690: 670 unit uPSI_StStrs;                            //SysTools
25691: 671 unit uPSI_StTree;                             //SysTools
25692: 672 unit uPSI_StVArr;                            //SysTools
25693: 673 unit uPSI_StRegIni;                            //SysTools
25694: 674 unit uPSI_urkf;                               //DMath
25695: 675 unit uPSI_usvd;                               //DMath
25696: 676 unit uPSI_DepWalkUtils;                         //JCL

```

```

25697: 677 unit uPSI_OptionsFrm;                                //JCL
25698: 678 unit yuvconverts;                                 //mX4
25699: 679 uPSI_JvPropAutoSave;                            //JCL
25700: 680 uPSI_AclAPI;                                    //alcinoe
25701: 681 uPSI_AviCap;                                    //alcinoe
25702: 682 uPSI_ALAVLBinaryTree;                           //alcinoe
25703: 683 uPSI_ALFcMisc;                                 //alcinoe
25704: 684 uPSI_ALStringList;                            //alcinoe
25705: 685 uPSI_ALQuickSortList;                           //alcinoe
25706: 686 uPSI_ALStaticText;                            //alcinoe
25707: 687 uPSI_ALJSONDoc;                               //alcinoe
25708: 688 uPSI_ALGSMComm;                             //alcinoe
25709: 689 uPSI_ALWindows;                                //alcinoe
25710: 690 uPSI_ALMultiPartFormDataParser;                //alcinoe
25711: 691 uPSI_ALHttpCommon;                            //alcinoe
25712: 692 uPSI_ALWebspider;                            //alcinoe
25713: 693 uPSI_ALHttpClient;                           //alcinoe
25714: 694 uPSI_ALFcHTML;                                //alcinoe
25715: 695 uPSI_ALFTPClient;                            //alcinoe
25716: 696 uPSI_ALInternetMessageCommon;                //alcinoe
25717: 697 uPSI_ALWininetHttpClient;                   //alcinoe
25718: 698 uPSI_ALWinInetFTPCClient;                  //alcinoe
25719: 699 uPSI_ALWinHttpWrapper;                      //alcinoe
25720: 700 uPSI_ALWinHttpClient;                        //alcinoe
25721: 701 uPSI_ALFcWinSock;                            //alcinoe
25722: 702 uPSI_ALFcSQL;                                //alcinoe
25723: 703 uPSI_ALFcCGI;                                //alcinoe
25724: 704 uPSI_ALFcExecute;                            //alcinoe
25725: 705 uPSI_ALFcFile;                                //alcinoe
25726: 706 uPSI_ALFcMime;                                //alcinoe
25727: 707 uPSI_ALPhpRunner;                            //alcinoe
25728: 708 uPSI_ALGraphic;                             //alcinoe
25729: 709 uPSI_ALIniFiles;                            //alcinoe
25730: 710 uPSI_ALMemCachedClient;                     //alcinoe
25731: 711 unit uPSI_MyGrids;                            //mX4
25732: 712 uPSI_ALMultiPartMixedParser;                 //alcinoe
25733: 713 uPSI_ALSMTPClient;                           //alcinoe
25734: 714 uPSI_ALNNTPClient;                           //alcinoe
25735: 715 uPSI_ALHintBalloon;                          //alcinoe
25736: 716 unit uPSI_ALXmlDoc;                         //alcinoe
25737: 717 unit uPSI_IPCThrd;                           //VCL
25738: 718 unit uPSI_MonForm;                            //VCL
25739: 719 unit uPSI_TeCanvas;                           //Orpheus
25740: 720 unit uPSI_Ovcmisc;                           //Orpheus
25741: 721 unit uPSI_ovcfiler;                          //Orpheus
25742: 722 unit uPSI_ovcstate;                          //Orpheus
25743: 723 unit uPSI_ovccoco;                           //Orpheus
25744: 724 unit uPSI_ovcrvexp;                          //Orpheus
25745: 725 unit uPSI_OvcFormatSettings;                //Orpheus
25746: 726 unit uPSI_OvcUtils;                          //Orpheus
25747: 727 unit uPSI_ovcstore;                          //Orpheus
25748: 728 unit uPSI_ovcstr;                            //Orpheus
25749: 729 unit uPSI_ovcmru;                            //Orpheus
25750: 730 unit uPSI_ovccmd;                            //Orpheus
25751: 731 unit uPSI_ovctimer;                          //Orpheus
25752: 732 unit uPSI_ovcintl;                           //Orpheus
25753: 733 uPSI_AfCircularBuffer;                     //AsyncFree
25754: 734 uPSI_AfUtils;                                //AsyncFree
25755: 735 uPSI_AfSafeSync;                            //AsyncFree
25756: 736 uPSI_AfComPortCore;                          //AsyncFree
25757: 737 uPSI_AfComPort;                            //AsyncFree
25758: 738 uPSI_AfPortControls;                        //AsyncFree
25759: 739 uPSI_AfDataDispatcher;                     //AsyncFree
25760: 740 uPSI_AfViewers;                             //AsyncFree
25761: 741 uPSI_AfDataTerminal;                        //AsyncFree
25762: 742 uPSI_SimplePortMain;                        //AsyncFree
25763: 743 unit uPSI_ovcclock;                          //Orpheus
25764: 744 unit uPSI_o32intlist;                      //Orpheus
25765: 745 unit uPSI_o32ledlabel;                     //Orpheus
25766: 746 unit uPSI_AlMySqlClient;                   //alcinoe
25767: 747 unit uPSI_ALFBXClient;                     //alcinoe
25768: 748 unit uPSI_ALFcSQL;                           //alcinoe
25769: 749 unit uPSI_AsyncTimer;                        //mX4
25770: 750 unit uPSI_ApplicationFileIO;                //mX4
25771: 751 unit uPSI_PsAPI;                            //VCLé
25772: 752 uPSI_ovcurl;                                //Orpheus
25773: 753 uPSI_ovcurl;                                //Orpheus
25774: 754 uPSI_ovcvlib;                              //Orpheus
25775: 755 uPSI_ovccolor;                            //Orpheus
25776: 756 uPSI_ALFBXLib;                            //alcinoe
25777: 757 uPSI_ovcmeter;                            //Orpheus
25778: 758 uPSI_ovcpeakm;                            //Orpheus
25779: 759 uPSI_O32BGSty;                            //Orpheus
25780: 760 uPSI_ovcBidi;                                //Orpheus
25781: 761 uPSI_ovctcarry;                           //Orpheus
25782: 762 uPSI_DXPUtil;                             //mX4
25783: 763 uPSI_ALMultiPartBaseParser;                 //alcinoe
25784: 764 uPSI_ALMultiPartAlternativeParser;          //alcinoe
25785: 765 uPSI_ALPOP3Client;                          //alcinoe

```

```

25786: 766 uPSI_SmallUtils;                                //mX4
25787: 767 uPSI_MakeApp;                                 //mX4
25788: 768 uPSI_O32MouseMon;                            //Orpheus
25789: 769 uPSI_OvcCache;                               //Orpheus
25790: 770 uPSI_ovccalc;                               //Orpheus
25791: 771 uPSI_Joystick;                             //OpenGL
25792: 772 uPSI_ScreenSaver;                           //OpenGL
25793: 773 uPSI_XCollection;                          //OpenGL
25794: 774 uPSI_Polynomials;                          //OpenGL
25795: 775 uPSI_PersistentClasses, //9.86           //OpenGL
25796: 776 uPSI_VectorLists;                           //OpenGL
25797: 777 uPSI_XOpenGL;                             //OpenGL
25798: 778 uPSI_MeshUtils;                           //OpenGL
25799: 779 unit uPSI_JclSysUtils;                   //JCL
25800: 780 unit uPSI_JclBorlandTools;                //JCL
25801: 781 unit JclFileUtils_max;                   //JCL
25802: 782 uPSI_AfDataControls;                     //AsyncFree
25803: 783 uPSI_GLSilhouette;                        //OpenGL
25804: 784 uPSI_JclSysUtils_class;                  //JCL
25805: 785 uPSI_JclFileUtils_class;                 //JCL
25806: 786 uPSI_FileUtil;                            //JCL
25807: 787 uPSI_changefind;                           //mX4
25808: 788 uPSI_cmdIntf;                            //mX4
25809: 789 uPSI_fservice;                           //mX4
25810: 790 uPSI_Keyboard;                            //OpenGL
25811: 791 uPSI_VRMLParser;                          //OpenGL
25812: 792 uPSI_GLFileVRML;                         //OpenGL
25813: 793 uPSI_Octree;                            //OpenGL
25814: 794 uPSI_GLPolyhedron;                       //OpenGL
25815: 795 uPSI_GLCrossPlatform;                   //OpenGL
25816: 796 uPSI_GLParticles;                        //OpenGL
25817: 797 uPSI_GLNavigator;                        //OpenGL
25818: 798 uPSI_GLStarRecord;                      //OpenGL
25819: 799 uPSI_GLTextureCombiners;                //OpenGL
25820: 800 uPSI_GLCanvas;                           //OpenGL
25821: 801 uPSI_GeometryBB;                         //OpenGL
25822: 802 uPSI_GeometryCoordinates;               //OpenGL
25823: 803 uPSI_VectorGeometry;                    //OpenGL
25824: 804 uPSI_BumpMapping;                        //OpenGL
25825: 805 uPSI_TGA;                               //OpenGL
25826: 806 uPSI_GLVectorFileObjects;              //OpenGL
25827: 807 uPSI_IMM;                               //VCL
25828: 808 uPSI_CategoryButtons;                  //VCL
25829: 809 uPSI_ButtonGroup;                      //VCL
25830: 810 uPSI_DbExcept;                          //VCL
25831: 811 uPSI_AxCtrls;                           //VCL
25832: 812 uPSI_GL_actorUnit1;                   //OpenGL
25833: 813 uPSI_StdVCL;                           //VCL
25834: 814 unit CurvesAndSurfaces;                //OpenGL
25835: 815 uPSI_DataAwareMain;                   //AsyncFree
25836: 816 uPSI_TabNotBk;                          //VCL
25837: 817 uPSI_udwsfiler;                        //mX4
25838: 818 uPSI_synaip;                           //Synapse!
25839: 819 uPSI_synacode;                         //Synapse
25840: 820 uPSI_synachar;                         //Synapse
25841: 821 uPSI_synamisc;                        //Synapse
25842: 822 uPSI_synaser;                          //Synapse
25843: 823 uPSI_synaicnv;                        //Synapse
25844: 824 uPSI_tlintsend;                        //Synapse
25845: 825 uPSI_pingsend;                         //Synapse
25846: 826 uPSI_blksock;                          //Synapse
25847: 827 uPSI_asnlutil;                         //Synapse
25848: 828 uPSI_dnssend;                          //Synapse
25849: 829 uPSI_clamsend;                         //Synapse
25850: 830 uPSI_ldapsend;                         //Synapse
25851: 831 uPSI_mimemess;                        //Synapse
25852: 832 uPSI_slogsend;                         //Synapse
25853: 833 uPSI_mimepart;                         //Synapse
25854: 834 uPSI_mimeinln;                         //Synapse
25855: 835 uPSI_ftpsend;                          //Synapse
25856: 836 uPSI_ftptsend;                         //Synapse
25857: 837 uPSI_httpsend;                         //Synapse
25858: 838 uPSI_sntpsend;                         //Synapse
25859: 839 uPSI_smtpsend;                         //Synapse
25860: 840 uPSI_snmpsend;                         //Synapse
25861: 841 uPSI_imapsend;                         //Synapse
25862: 842 uPSI_pop3send;                         //Synapse
25863: 843 uPSI_nntpsend;                         //Synapse
25864: 844 uPSI_ssl_cryptlib;                   //Synapse
25865: 845 uPSI_ssl_openssl;                   //Synapse
25866: 846 uPSI_synhttp_daemon;                 //Synapse
25867: 847 uPSI_NetWork;                          //mX4
25868: 848 uPSI_PingThread;                      //Synapse
25869: 849 uPSI_JvThreadTimer;                   //JCL
25870: 850 unit uPSI_wwSystem;                   //InfoPower
25871: 851 unit uPSI_IdComponent;                //Indy
25872: 852 unit uPSI_IdIOHandlerThrottle;       //Indy
25873: 853 unit uPSI_Themes;                     //VCL
25874: 854 unit uPSI_StdStyleActnCtrls;         //VCL

```

```

25875: 855 unit uPSI_UDDIHelper; //VCL
25876: 856 unit uPSI_IdIMAP4Server; //Indy
25877: 857 uPSI_VariantSymbolTable; //VCL //3.9.9.92
25878: 858 uPSI_udf_glob; //mX4
25879: 859 uPSI_TabGrid; //VCL
25880: 860 uPSI_JsDBTreeView; //mX4
25881: 861 uPSI_JsSendMail; //mX4
25882: 862 uPSI_dbTvRecordList; //mX4
25883: 863 uPSI_TreeVwEx; //mX4
25884: 864 uPSI_ECDataLink; //mX4
25885: 865 uPSI_dbTree; //mX4
25886: 866 uPSI_dbTreeCBox; //mX4
25887: 867 unit uPSI_Debug; //TfrmDebug //mX4
25888: 868 uPSI_TimeFncs; //mX4
25889: 869 uPSI_FileIntf; //VCL
25890: 870 uPSI_SockTransport; //RTL
25891: 871 unit uPSI_WinInet; //RTL
25892: 872 unit uPSI_Wwstr; //mX4
25893: 873 uPSI_DBLookup; //VCL
25894: 874 uPSI_Hotspot; //mX4
25895: 875 uPSI_HList; //History List //mX4
25896: 876 unit uPSI_DrTable; //VCL
25897: 877 uPSI_TConnect; //VCL
25898: 878 uPSI_DataBkr; //VCL
25899: 879 uPSI_HTTPIntr; //VCL
25900: 880 unit uPSI_Mathbox; //mX4
25901: 881 uPSI_cyIndy; //cY
25902: 882 uPSI_cySysUtils; //cY
25903: 883 uPSI_cyWinUtils; //cY
25904: 884 uPSI_cyStrUtils; //cY
25905: 885 uPSI_cyObjUtils; //cY
25906: 886 uPSI_cyDateUtils; //cY
25907: 887 uPSI_cyBDE; //cY
25908: 888 uPSI_cyClasses; //cY
25909: 889 uPSI_cyGraphics; //3.9.9.94_2 //cY
25910: 890 unit uPSI_cyTypes; //cY
25911: 891 uPSI_JvDateTimePicker; //JCL
25912: 892 uPSI_JvCreateProcess; //JCL
25913: 893 uPSI_JvEasterEgg; //JCL
25914: 894 uPSI_WinSvc; //3.9.9.94_3 //VCL
25915: 895 uPSI_SvcMgr; //VCL
25916: 896 unit uPSI_JvPickDate; //JCL
25917: 897 unit uPSI_JvNotify; //JCL
25918: 898 uPSI_JvStrHlder; //JCL
25919: 899 unit uPSI_JclNTFS2; //JCL
25920: 900 uPSI_Jcl18087 //math coprocessor //JCL
25921: 901 uPSI_JvAddPrinter //JCL
25922: 902 uPSI_JvCabFile //JCL
25923: 903 uPSI_JvDataEmbedded; //JCL
25924: 904 unit uPSI_U_HexView; //mX4
25925: 905 uPSI_UWavein4; //mX4
25926: 906 uPSI_AMixer; //mX4
25927: 907 uPSI_JvaScrollText; //mX4
25928: 908 uPSI_JvArrow; //mX4
25929: 909 unit uPSI_UrlMon; //mX4
25930: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
25931: 911 unit uPSI_U_Oscilloscope4; //TosfrmMain; //DFF
25932: 912 unit uPSI_DFFUtils; //DFF
25933: 913 unit uPSI_MathsLib; //DFF
25934: 914 uPSI_UIntList; //DFF
25935: 915 uPSI_UGetParens; //DFF
25936: 916 unit uPSI_UGeometry; //DFF
25937: 917 unit uPSI_UAstronomy; //DFF
25938: 918 unit uPSI_UCardComponentV2; //DFF
25939: 919 unit uPSI_UTGraphSearch; //DFF
25940: 920 unit uPSI_UParser10; //DFF
25941: 921 unit uPSI_cyIEUtils; //cY
25942: 922 unit uPSI_UcomboV2; //DFF
25943: 923 uPSI_cyBaseComm; //cY
25944: 924 uPSI_cyAppInstances; //cY
25945: 925 uPSI_cyAttract; //cY
25946: 926 uPSI_cyDERUtils //cY
25947: 927 unit uPSI_cyDocER; //cY
25948: 928 unit uPSI_ODBC; //mX
25949: 929 unit uPSI_AssocExec; //mX
25950: 930 uPSI_cyBaseCommRoomConnector; //cY
25951: 931 uPSI_cyCommRoomConnector; //cY
25952: 932 uPSI_cyCommunicate; //cY
25953: 933 uPSI_cyImage; //cY
25954: 934 uPSI_cyBaseContainer; //cY
25955: 935 uPSI_cyModalContainer; //cY
25956: 936 uPSI_cyFlyingContainer; //cY
25957: 937 uPSI_RegStr; //VCL
25958: 938 uPSI_HtmlHelpViewer; //VCL
25959: 939 unit uPSI_cyIniForm //cY
25960: 940 unit uPSI_cyVirtualGrid; //cY
25961: 941 uPSI_Profiler; //DA
25962: 942 uPSI_BackgroundWorker; //DA
25963: 943 uPSI_WavePlay; //DA

```

```

25964: 944 uPSI_WaveTimer, //DA
25965: 945 uPSI_WaveUtils; //DA
25966: 946 uPSI_NamedPipes, //TB
25967: 947 uPSI_NamedPipeServer, //TB
25968: 948 unit uPSI_process, //TB
25969: 949 unit uPSI_DPUtils; //TB
25970: 950 unit uPSI_CommonTools; //TB
25971: 951 uPSI_DataSendToWeb, //TB
25972: 952 uPSI_StarCalc, //TB
25973: 953 uPSI_D2_XPVistaHelperU //TB
25974: 954 unit uPSI_NetTools //TB
25975: 955 unit uPSI_Pipes //TB
25976: 956 uPSI_ProcessUnit, //mX
25977: 957 uPSI_adGSM, //mX
25978: 958 unit uPSI_BetterADODataSet; //mX
25979: 959 unit uPSI_AdSelCom; //FTT //mX
25980: 960 unit unit uPSI_dwsXPlatform; //DWS
25981: 961 uPSI_AdSocket; //mX Turbo Power
25982: 962 uPSI_AdPacket; //mX
25983: 963 uPSI_AdPort; //mX
25984: 964 uPSI_PathFunc; //Inno
25985: 965 uPSI_CmnFunc; //Inno
25986: 966 uPSI_CmnFunc2; //Inno Setup //Inno
25987: 967 unit uPSI_BitmapImage; //mX4
25988: 968 unit uPSI_ImageGrabber; //mX4
25989: 969 uPSI_SecurityFunc; //Inno
25990: 970 uPSI_RedirFunc; //Inno
25991: 971 uPSI_FIFO, (MemoryStream) //mX4
25992: 972 uPSI_Int64Fm; //Inno
25993: 973 unit uPSI_InstFunc; //Inno
25994: 974 unit uPSI_LibFusion; //Inno
25995: 975 uPSI_SimpleExpression; //Inno
25996: 976 uPSI_unitResourceDetails; //XN
25997: 977 uPSI_unitResFile; //XN
25998: 978 unit uPSI_simpleComport; //mX4
25999: 979 unit uPSI_AfViewershelpers; //Async
26000: 980 unit uPSI_Console; //mX4
26001: 981 unit uPSI_AnalogMeter; //TB
26002: 982 unit uPSI_XPrinter; //TB
26003: 983 unit uPSI_IniFiles; //VCL
26004: 984 unit uPSI_lazIniFiles; //FP
26005: 985 uPSI_testutils; //FP
26006: 986 uPSI_ToolsUnit; (DBTests) //FP
26007: 987 uPSI_fpcunit //FP
26008: 988 uPSI_testdecorator; //FP
26009: 989 unit uPSI_fpcunittests; //FP
26010: 990 unit uPSI_cTCPBuffer; //Fundamentals 4
26011: 991 unit uPSI_Glut; //Open GL
26012: 992 uPSI_LEDBitmaps; //mX4
26013: 993 uPSI_FileClass; //Inno
26014: 994 uPSI_FileUtilsClass; //mX4
26015: 995 uPSI_ComPortInterface; //Kit //mX4
26016: 996 unit uPSI_SwitchLed; //mX4
26017: 997 unit uPSI_cyDmmCanvas; //cY
26018: 998 uPSI_uColorFunctions; //DFF
26019: 999 uPSI_uSettings; //DFF
26020: 1000 uPSI_cyDebug.pas //cY
26021: 1001 uPSI_cyColorMatrix; //cY
26022: 1002 unit uPSI_cyCopyFiles; //cY
26023: 1003 unit uPSI_cySearchFiles; //cY
26024: 1004 unit uPSI_cyBaseMeasure; //cY
26025: 1005 unit uPSI_PJISStreams; //DD
26026: 1006 unit uPSI_cyRunTimeResize; //cY
26027: 1007 unit uPSI_jcontrolutils; //cY
26028: 1008 unit uPSI_kcMapView; (+GEONames) //kc
26029: 1009 unit uPSI_kcMapViewDESynapse; //kc
26030: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
26031: 1011 unit uPSI_LedNumber; //TurboPower
26032: 1012 unit uPSI_StStrL; //SysTools
26033: 1013 unit uPSI_indGnouMeter; //LAZ
26034: 1014 unit uPSI_Sensors; //LAZ
26035: 1015 unit uPSI_pwmain; //cgi of powtills //Pow
26036: 1016 unit uPSI_HTMLUtil; //Pow
26037: 1017 unit uPSI_synwrap1; //httpsend //Pow
26038: 1018 unit StreamWrap1 //Pow
26039: 1019 unit uPSI_pwmain; //Pow
26040: 1020 unit pwtypes //Pow
26041: 1021 uPSI_W32VersionInfo //LAZ
26042: 1022 unit uPSI_IpAnim; //LAZ
26043: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
26044: 1024 unit uPSI_LrtPoTools; //LAZ
26045: 1025 unit uPSI_Laz_DOM; //LAZ
26046: 1026 unit uPSI_hhAvComp; //LAZ
26047: 1027 unit uPSI_GPS2; //mX4
26048: 1028 unit uPSI_GPS; //mX4
26049: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo //mX4
26050: 1030 unit uPSI_NMEA; // GPS //mX4
26051: 1031 unit uPSI_ScreenThreeDLab; //mX4
26052: 1032 unit uPSI_Spin; //VCL

```

```

26053: 1033 unit uPSI_DynaZip; //mX4
26054: 1034 unit uPSI_clockExpert; //TB
26055: 1035 unit debugLn //mX4
26056: 1036 uPSI_SortUtils; //Jcl
26057: 1037 uPSI_BitmapConversion; //Jcl
26058: 1038 unit uPSI_JclTD32; //Jcl
26059: 1039 unit uPSI_ZDbcUtils; //Zeos
26060: 1040 unit uPSI_ZScriptParser; //Zeos
26061: 1041 uPSI_JvIni; //JCL
26062: 1042 uPSI_JvFtpGrabber; //JCL
26063: 1043 unit uPSI_NeuralNetwork; //OCL
26064: 1044 unit uPSI_StExpr; //SysTools
26065: 1045 unit uPSI_GR32_Geometry; //GR32
26066: 1046 unit uPSI_GR32_Containers; //GR32
26067: 1047 unit uPSI_GR32_Backends_VCL; //GR32
26068: 1048 unit uPSI_STSaturn; //Venus+Mercury+Mars++ //SysTools
26069: 1049 unit uPSI_JclParseUses; //JCL
26070: 1050 unit uPSI_JvFinalize; //JCL
26071: 1051 unit uPSI_panUnit1; //GLScene
26072: 1052 unit uPSI_DD83ul; //Arduino Tester //mX4
26073: 1053 unit uPSI_BigIni //Hinzen
26074: 1054 unit uPSI_ShellCtrls; //VCL
26075: 1055 unit uPSI_fmath; //FMath
26076: 1056 unit uPSI_fComp; //FMath
26077: 1057 unit uPSI_HighResTimer; //Lauer
26078: 1058 unit uconvMain; //Unit Converter //PS
26079: 1059 unit uPSI_uconvMain; //PS
26080: 1060 unit uPSI_ParserUtils; //PS
26081: 1061 unit uPSI_UPSUtils; //PS
26082: 1062 unit uPSI_ParserU; //PS
26083: 1063 unit uPSI_TypeInfo; //VCL
26084: 1064 unit uPSI_ServiceMgr; //mX
26085: 1065 unit uPSI_UDict; //DFP
26086: 1066 unit uPSI_ubigFloatV3; //DFP
26087: 1067 unit uPSI_UBigIntsV4; //DFP
26088: 1068 unit uPSI_ServiceMgr2; //mX
26089: 1069 unit uPSI_UP10Build; //PS
26090: 1070 unit uPSI_UParser10; //PS
26091: 1071 unit uPSI_IdModBusServer; //MB
26092: 1072 unit uPSI_IdModBusClient; //MB
26093: 1073 unit uPSI_ColorGrd; //VCL
26094: 1074 unit uPSI_DirOutln; //VCL
26095: 1075 unit uPSI_Gauges; //VCL
26096: 1076 unit uPSI_CustomizeDlg; //VCL
26097: 1077 unit uPSI_ActnMan; //VCL
26098: 1078 unit uPSI_CollPanl; //VCL
26099: 1079 unit uPSI_Calendar2; //VCL
26100: 1080 unit uPSI_IBCtrls; //VCL
26101: 1081 unit uPSI_IdStackWindows; //Indy
26102: 1082 unit uPSI_CTSVendorUtils; //DBX
26103: 1083 unit uPSI_VendorTestFramework; //DBX
26104: 1084 unit uPSI_TInterval; //mX4
26105: 1085 unit uPSI_JvAnimate; //JCL
26106: 1086 unit uPSI_DBXCharDecoder; //DBX
26107: 1087 unit uPSI_JvDBLists; //JCL
26108: 1088 unit uPSI_JvFileInfo; //JCL
26109: 1089 unit uPSI_SOAPConn; //VCL
26110: 1090 unit uPSI_SOAPLinked; //VCL
26111: 1091 unit uPSI_XSBuiltIns; //VCL
26112: 1092 unit uPSI_JvgDigits; //JCL
26113: 1093 unit uPSI_JvDesignUtils; //JCL
26114: 1094 unit uPSI_JvgCrossTable; //JCL
26115: 1095 unit uPSI_JvgReport; //JCL
26116: 1096 unit uPSI_JvDBRichEdit; //JCL
26117: 1097 unit uPSI_JvWinHelp; //JCL
26118: 1098 unit uPSI_WaveConverter; //JCL
26119: 1099 unit uPSI_ACMDconvertor; //JCL
26120: 1100 unit XSBuiltIns_Routines //JCL
26121: 1101 unit uPSI_ComObjOleDB_utils.pas //JCL
26122: 1102 unit uPSI_SMScript; //JCL
26123: 1103 unit uPSI_CompFileIo; //JCL
26124: 1104 unit uPSI_SynHighlighterGeneral; //JCL
26125: 1105 unit uPSI_geometry2; //JCL
26126: 1106 unit uPSI_MConnect; //JCL
26127: 1107 unit uPSI_ObjBrkr; //JCL
26128: 1108 unit uPSI_uMultiStr; //JCL
26129: 1109 unit uPSI_WinAPI.pas; //JCL
26130: 1110 unit uPSI_JvAVICapture; //JCL
26131: 1111 unit uPSI_JvExceptionForm; //JCL
26132: 1112 unit uPSI_JvConnectNetwork; //JCL
26133: 1113 unit uPSI_MTM MainForm; //JCL
26134: 1114 unit uPSI_DdeMan; //JCL
26135: 1115 unit uPSI_DIUutils; //JCL
26136: 1116 unit uPSI_gnugettext; //JCL
26137: 1117 unit uPSI_Xmlxform; //JCL
26138: 1118 unit uPSI_SvrHTTPIndy; //JCL
26139: 1119 unit uPSI_CPortTrmSet; //JCL
26140: 1120 unit SvrLog; -----
26141: 1121 unit uPSI_IndySockTransport.pas (+IdHTTPHeaderInfo) //mX4

```

```
26142: 1122 unit uPSI_HTTPProd.pas
26143: 1123 unit uPSI_CppParser.pas
26144: 1124 unit uPSI_SynHighlighterCpp.pas
26145: 1125 unit uPSI_CodeCompletion.pas
26146: 1126 unit uPSI_U_IntList2.pas
26147: 1127 unit uPSI_SockHTTP.pas
26148: 1128 uPSI_SockAppNotify.pas
26149: 1129 uPSI_NSToIS.pas
26150: 1130 unit uPSI_DBOLECtl.pas
26151: 1131 unit uPSI_xercesxmlDom;
26152: 1132 unit uPSI_xmldom;
26153: 1133 unit uPSI_Midas;
26154: 1134 unit uPSI_JclExprEval;
26155: 1135 uPSI_Gameboard;
26156: 1136 unit uPSI_ExtUtil;
26157: 1137 unit uPSI_FCGIApp;
26158: 1138 unit uPSI_ExtPascal;
26159: 1139 unit uPSI_PersistSettings;
26160: 1140 IdHTTPHeaderInfo.pas
26161: 1141 uPSI_SynEditAutoComplete;
26162: 1142 uPSI_SynEditTextBuffer.pas
26163: 1143 unit uPSI_JclPCRE;
26164: 1144 unit uPSI_ZConnection;
26165: 1145 unit uPSI_ZSequence;
26166: 1146 unit uPSI_ChessPrg;
26167: 1147 unit uPSI_ChessBrd;
26168: 1148 unit uPSI_Graph3D;
26169: 1149 uPSI_SysInfoCtrls2.pas
26170: 1150 unit uPSI_RegUtils;
26171: 1151 unit uPSI_VariantRtn;
26172: 1152 uPSI_Stdfuncs,
26173: 1153 unit uPSI_SqlTxtRtns;
26174: 1154 unit uPSI_BSpectrum;
26175: 1155 unit IPAddressControl;
26176: 1156 unit uPSI_Paradox;
26177: 1157 unit uPSI_Environ;
26178: 1158 uPSI_GraphicsPrimitivesLibrary;
26179: 1159 uPSI_DrawFigures,
26180: 1160 unit uPSI_synadbg;
26181: 1161 unit uPSI_BitStream;
26182: 1162 unit uPSI_xrtl_util_FileVersion;
26183: 1163 uPSI_XmlRpcCommon,
26184: 1164 unit uPSI_XmlRpcClient;
26185: 1165 unit uPSI_XmlRpcTypes;
26186: 1166 unit uPSI_XmlRpcServer;
26187: 1167 unit uPSI_SynAutoIndent;
26188: 1168 unit uPSI_synafpc;
26189: 1169 unit uPSI_RxNotify;
26190: 1170 unit uPSI_SynAutoCorrect;
26191: 1171 unit uPSI_rxOle2Auto;
26192: 1172 unit uPSI_Spring_Utilsmx;
26193: 1173 unit uPSI_ulogifit;
26194: 1174 unit uPSI_HarmFade;
26195: 1175 unit uPSI_SynCompletionProposal;
26196: 1176 unit uPSI_rxAniFile;
26197: 1177 uPSI_ulinfilt,
26198: 1178 uPSI_usvdfit;
26199: 1179 unit uPSI_JclStringLists;
26200: 1180 unit uPSI_ZLib;
26201: 1181 unit uPSI_MaxTokenizers; //WANT
26202: 1182 unit uPSI_MaxStrUtils;
26203: 1183 unit uPSI_MaxXMLUtils;
26204: 1184 unit uPSI_MaxUtils;
26205: 1185 unit uPSI_VListBox;
26206: 1186 unit uPSI_MaxDOM;
26207: 1187 unit uPSI_MaxDOMDictionary;
26208: 1188 unit uPSI_MaxDOMDictionary_Routines;
26209: 1189 unit uPSI_cASN1;
26210: 1190 unit uPSI_cX509Certificate;
26211: 1191 unit uPSI_uciaXml;
26212: 1192 unit uPSI.StringsW;
26213: 1193 unit uPSI_FilestreamW; //WideString D7X
26214: 1194 unit Drawingutils;
26215: 1195 unit uPSI_InetUtilsUnified;
26216: 1196 unit uPSI_FileMask;
26217: 1197 unit uPSI_StrConv;
26218: 1198 unit uPSI_Simpat;
26219: 1199 unit uPSI_Tooltips.pas
26220: 1200 unit uPSI_StringGridLibrary.pas
26221: 1201 unit uPSI_ChronCheck.pas
26222: 1202 unit uPSI_REXX.pas
26223: 1203 uPSI_SysImg.pas
26224: 1204 unit uPSI_Tokens;
26225: 1205 unit uPSI_KFunctions;
26226: 1206 unit uPSI_KMessageBox;
26227: 1207 unit uPSI_CPUSpeed.pas
26228: 1208 uPSI_RoboTracker.pas
26229: 1209 unit uPSI_NamedPipesImpl.pas
26230: 1210 unit uPSI_KLog.pas
```

```

26231: 1211 unit uPSI_NamedPipeThreads.pas
26232: 1212 unit uPSI_MapFiles.pas //map stream of memory-mapped files
26233: 1213 unit uPSI_BKPwdGen, //Password Generator
26234: 1214 unit uPSI_Kronos, // big chrono date time library
26235: 1215 unit uPSI_TokenLibrary2;
26236: 1216 uPSI_KDialogs,
26237: 1217 uPSI_Numedit,
26238: 1218 unit uPSI_StSystem2;
26239: 1219 unit uPSI_KGraphics;
26240: 1220 uPSI_KGraphics_functions;
26241: 1221 uPSI_umaxPipes.pas
26242: 1222 unit uPSI_KControls;
26243: 1223 unit SysUtils_max2;
26244: 1224 uPSI_IdAntiFreeze.pas
26245: 1225 uPSI_IdLogStream.pas
26246: 1226 unit uPSI_IdThreadsafe;
26247: 1227 unit uPSI_IdThreadMgr;
26248: 1228 unit uPSI_IdAuthentication;
26249: 1229 unit uPSI_IdAuthenticationManager;
26250: 1230 uPSI_OverbyteIcsConApp
26251: 1231 unit uPSI_KMemo;
26252: 1232 unit uPSI_OverbyteIcsTicks64;
26253: 1233 unit uPSI_OverbyteIcsShal.pas
26254: 1234 unit uPSI_KEditCommon.pas
26255: 1235 unit uPSI_UtilsMax4.pas
26256: 1236 unit uPSI_IdNNTPServer;
26257: 1237 unit uPSI_UWANTUtils;
26258: 1238 unit uPSI_UtilsMax5.pas;
26259: 1239 unit uPSI_OverbyteIcsAsn1Utils;
26260: 1240 unit uPSI_IdHTTPHeaderInfo; //mx4 response headers
26261: 1241 uPSI_wmiserv.pas {uPSI_SimpleSFTP.pas}
26262: 1242 uPSI_WbemScripting_TLB.pas
26263: 1243 unit uPSI_uJSON;
26264: 1244 uPSI_RegSvrUtils.pas
26265: 1245 unit uPSI_osfileUtil;
26266: 1246 unit uPSI_SHDocVw; //TWebbrowser
26267: 1247 unit uPSI_SHDocVw_TLB
26268: 1248 uPSC_classes.pas V2
26269: 1249 uPSR_classes.pas V2
26270: 1250 uPSI_U_Oscilloscope4_2
26271: 1251 uPSI_xutils.pas
26272: 1252 uPSI_ietf.pas
26273: 1253 uPSI_iso3166.pas
26274: 1254 uPSI_dateutil_real.pas //Optima ISO 8601
26275: 1255 uPSI_dateext4.pas
26276: 1256 uPSI_locale.pas //ISO Formater & Tests
26277: 1257 file charset.inc //IANA Registered character sets
26278: 1258 unit uPSI.Strings;
26279: 1259 unit uPSI_crc_checks; //ISO 3309 and ITU-T-V42
26280: 1260 unit uPSI_extDOS;
26281:
26282: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
26283:
26284:
26285: http://www.slideshare.net/maxkleinerl/codereview-topics
26286: /////////////////////////////////
26287: //Form Template Library FTL
26288: /////////////////////////////////
26289:
26290: FTL For Form Building Lib out of the Script, eg. 399 form_templates.txt
26291: 045 unit uPSI_VListView TFormListView;
26292: 263 unit uPSI_JvProfiler32; TProfReport
26293: 270 unit uPSI_ImgList; TCustomImageList
26294: 278 unit uPSI_JvImageWindow; TJvImageWindow
26295: 317 unit uPSI_JvParserForm; TJvHTMLParserForm
26296: 497 unit uPSI_DebugBox; TDebugBox
26297: 513 unit uPSI_ImageWin; TImageForm, TImageForm2
26298: 514 unit uPSI_CustomDrawTreeView; TCustomDrawForm
26299: 515 unit uPSI_GraphWin; TGraphWinForm
26300: 516 unit uPSI_actionMain; TActionForm
26301: 518 unit uPSI_CtlPanel; TAppletApplication
26302: 529 unit uPSI_MDIEdit; TEditForm //RichEditApp
26303: 535 unit uPSI_CoolMain; {browser} TWeb MainForm
26304: 538 unit uPSI_frmExportMain; TSynexportForm
26305: 585 unit uPSI_usniffer; //PortScanForm; TSniffForm
26306: 600 unit uPSI_ThreadForm; TThreadSortForm;
26307: 618 unit uPSI_delphi_arduino_Unit1; TLEDForm
26308: 620 unit uPSI_fplotMain; TfplotForm1
26309: 660 unit uPSI_JvDBQueryParamsForm; TJvQueryParamsDialog
26310: 677 unit uPSI_OptionsFrm; TfrmOptions;
26311: 718 unit uPSI_MonForm; TMonitorForm
26312: 742 unit uPSI_SimplePortMain; TPortForm1
26313: 770 unit uPSI_ovccalc; TOvcCalculator //widget
26314: 810 unit uPSI_DbExcept; TDbEngineErrorDlg
26315: 812 unit uPSI_GL_actorUnit1; TglActorForm1 //OpenGL Robot
26316: 846 unit uPSI_synhttp_daemon; TTCPHttpDaemon, TTCPHttpThrd, TPingThread
26317: 867 unit uPSI_Debug; TfrmDebug
26318: 901 unit uPSI_JvAddPrinter; TJvAddPrinter //JCL
26319: 904 unit uPSI_U_HexView; THexForm2

```

```

26320: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)           TOscfrmMain
26321: 959 unit uPSI_AdSelCom;                                TComSelectForm
26322: 1029 unit uPSI_GPSUDemo;                               TFDemo
26323: 1031 unit uPSI_ScreenThreeDLab;                          TFormLab3D
26324: 1051 unit uPSI_panUnit1;                                TPanForm1 //GLScene
26325: 1052 unit uPSI_DD83ul; {Arduino Tester Frm}          TDD83f1
26326: 1059 unit uPSI_uconvMain;                               TfconvMain //PS
26327: 1076 unit uPSI_CustomizeDlg;                           TCustomizeDlg / TJvAddPrinterDialog;
26328: 1111 unit uPSI_JvExceptionForm;                         TJvErrorDialog //ShowException
26329: 1113 unit uPSI_MTMainForm;                            TvtMainForm
26330: 1119 unit uPSI_CPortTrmSet;                           TComTrmSetForm
26331: 1146 unit uPSI_ChessPrg;                             TChessForm1
26332: 1216 uPSI_KDialogs,                                     TKBrowseFolderDialog
26333:
26334:
26335: FormTemplates with <Ctrl> J
26336:
26337: [myformtemplate | formtemplate statement | Borland.EditOptions.Pascal] //with
26338: [myFastForm | class declaration | Borland.EditOptions.Pascal] //Dialog Form
26339: [myForm | class declaration | Borland.EditOptions.Pascal] //with Events
26340: [aForm | class declaration | Borland.EditOptions.Pascal] //single Form
26341: [getForm(x,y)] of object TForm [loadForm(x,y)]
26342: [ticker | ticker statement | Borland.EditOptions.Pascal]
26343: [except | exception statement | Borland.EditOptions.Pascal]
26344:
26345: procedure SIRegister_JvDesignUtils(CL: TPSPascalCompiler);
26346: begin
26347:   Function DesignClientToParent(const APoint: TPoint; AControl, AParent: TControl) : TPoint';
26348:   Function DesignMin(AA, AB : Integer):Integer';
26349:   Function DesignMax(AA, AB: Integer): Integer';
26350:   Function DesignRectWidth( const ARect : TRect) : Integer';
26351:   Function DesignRectHeight( const ARect : TRect) : Integer';
26352:   Function DesignValidateRect( const ARect : TRect) : TRect';
26353:   Function DesignNameIsUnique( AOwner : TComponent; const AName : string) : Boolean';
26354:   Function DesignUniqueName( AOwner : TComponent; const AClassName : string) : string';
26355:   Procedure DesignPaintRubberbandRect(AContainer:TWinControl;ARect:TPRect;APenStyle:TPenStyle);
26356:   Procedure DesignPaintGrid(ACanvas:TCanvas;const
      ARect:TPRect;ABackClr:TColor;AGridClr:TColor;ADivPixels:Int)
26357:   Procedure DesignPaintRules(ACanvas:TCanvas;const ARect:TPRect;ADivPixels:Int;ASubDivs:Bool;
26358:   Procedure DesignSaveComponentToStream( AComp : TComponent; AStream : TStream)');
26359:   Function DesignLoadComponentFromStream(AComp:TComponent;AStream:TStream;AOnError:TReaderError):TComponent;
26360:   Procedure DesignSaveComponentToFile( AComp : TComponent; const AFileName : string)');
26361:   Procedure DesignLoadComponentFromFile(AComp:TComp;const AFileName:string;AOnError:TReaderError);
26362: end;
26363:
26364: ex.:with TEditForm.create(self) do begin
26365:   caption:='Template Form Tester';
26366:   FormStyle:= fsStayOnTop;
26367:   with editor do begin
26368:     Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mx2.rtf'
26369:     SelStart:= 0; Modified:= False;
26370:   end;
26371: end;
26372: with TWeb MainForm.create(self) do begin
26373:   URLs.Text:= 'http://www.kleiner.ch';
26374:   URLsClick(self); Show;
26375: end;
26376: with TSynexportForm.create(self) do begin
26377:   Caption:= 'Synexport HTML RTF tester';
26378:   Show;
26379: end;
26380: with TThreadSortForm.create(self) do begin
26381:   showmodal; free;
26382: end;
26383: with TCustomDrawForm.create(self) do begin
26384:   width:=820; height:=820;
26385:   image1.height:= 600; //add properties
26386:   image1.picture.bitmap:= image2.picture.bitmap;
26387:   //SelectionBackground1Click(self) CustomDraw1Click(self);
26388:   Background1.click;
26389:   bitmap1.click; Tile1.click;
26390:   Showmodal;
26391:   Free;
26392: end;
26393: with TfplotForm1.Create(self) do begin
26394:   BtnPlotClick(self);
26395:   Showmodal; Free;
26396: end;
26397: with TOvcCalculator.create(self) do begin
26398:   parent:= aForm; //free;
26399:   setbounds(550,510,200,150);
26400:   displaystr:= 'maXcalc';
26401: end;
26402: with THexForm2.Create(self) do begin
26403:   ShowModal;
26404:   Free;
26405: end;
26406:
26407: function CheckBox: string;

```

```

26408: var idHTTP: TIdHTTP;
26409: begin
26410:   result:= 'version not found';
26411:   if IsInternet then begin
26412:     idHTTP:= TIdHTTP.Create(NIL);
26413:     try
26414:       result:= idHTTP.Get(MXVERSIONFILE2);
26415:       result:= result[1]+result[2]+result[3]+result[4]+result[5];
26416:       if result = MBVER2 then begin
26417:         //Speak(' A new Version '+vstr+' of max box is available! ');
26418:         result:= ('!!!! OK. You have latest Version: '+result+ ' available at '+MXSITE);
26419:       end; //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
26420:     finally
26421:       idHTTP.Free
26422:     end;
26423:   end;
26424: end;
26425:
26426: //Runtimer Spec Functions Edition 190
26427: function ApWinExecAndWait32(FileName:PChar;CommandLine:PChar; Visibility:Integer):Integer;
26428: function KillTask(ExeFileName: string): Integer;
26429: procedure KillProcess(hWindowHandle: HWND);
26430: function FindwindowByTitle(WindowTitle: string): Hwnd;
26431: function OpenIE(aURL: string): boolean;
26432: function XRTLIsInMainThread: Boolean;
26433: function IsInMainThread: Boolean;
26434: function IntToFloat(i: Integer): double;
26435: function AddThousandsSeparator(S: string; myChr: Char): string;
26436: function mciSendString(cmd: PChar; ret: PChar; len: integer; callback: integer): cardinal;
26437: procedure FormAnimation(Sender: TObject; adelay: integer);
26438: procedure LoadResourceFile2(aFile:string; ms:TMemoryStream);
26439: function putBinResTo(binresname: pchar; newpath: string): boolean;
26440: procedure ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:string);
26441: function IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:string;MouseX,MouseY:Integer;var HyperLink:string):Bool
26442: Function GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD) : DWORD;')
26443: Function GetWindowTask( hWnd : HWND ) : THandle;';
26444: Function LoadBitmap( hInstance : HINST; lpBitmapName : PChar ) : HBITMAP;';
26445: Function GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL';
26446: function WinExecAndWait32(FileName: string; Visibility: Integer): Longword;
26447: Function MakeHash( const s : TbtString ) : Longint';
26448: Function GetUsedUnitList( list : TStringlist ) : string';
26449: function ConsoleCapture(const _dirName, _exeName, _cmdLine: string; amemo: TStringlist): Bool;
26450: function ConsoleCaptureDOS(const _dirName, _exeName, _cmdLine: string; amemo: TStrings): Bool;
26451:   srlist:= TStringlist.create;
26452:   ConsoleCapture('C:\', 'cmd.exe', '/c dir *.*',srlist);
26453:   writeln(srlist.text); srlist.Free;
26454: function RunCaptured(const _dirName, _exeName,_cmdLine:string; amemo:TStringlist):Bool;';
26455: Function SamePropTypeName( const Name1, Name2 : ShortString ) : Boolean';
26456: Function FloatToStrEx( Value : Extended ) : string';
26457: Function StrToFloatEx( const S : string ) : Extended';
26458: Procedure PerformanceDelayMS(ams: integer); //microsecond resolution delay!
26459: //http://www.swissdelphicenter.ch/en/showcode.php?id=2179
26460: function ExecuteProcess(FileName: string;Visibility:Integer; BitMask:Integer; Synch:Boolean):Longword;
26461: Function ExecuteMultiProcessor(FileName:string;Visibility:Int;BitMask:Int;Synch:Bool):Longword;
26462:   if ExecuteMultiProcessor('notepad.exe', SW_SHOW, 2, true) = 0 then
26463:     writeln('Multiprocessing Runs on CPU 2');
26464: procedure StartServiceAfterInstall(aserv: TService);
26465: Function GetDllVersion2(DllName: string; var DLLVersionInfo: TDLLVersionInfo): Boolean;
26466: procedure SendCopyMessage(amess, astation: string); //communicate process-spanned with WM_COPYDATA
26467: function BrowseComputer2(DialogTitle: string; var CompName: string; bNewStyle: Bool): Bool;
26468: function ChangeAlphaTo(input: string; aoffset: byte): string';
26469: function CheckIBAN(iban: string): Boolean;';
26470: function CreateIDStack; //instance to Idstack of CreateStack
26471: Function GetRecordCount(DataSet: TBDEDataSet): Longint;';
26472: Function CountPos(const subtxt: string; Text: string): Integer;
26473: procedure HTMIToRTF(html: string; var rtf: TRichedit);
26474: procedure ReversePlay(const szFileName: string);
26475: Function ADOConnectionString(ParentHandle:THandle;InitialString:WideString;out NewString:string):Bool;
26476: procedure ShowEOLEException(AExc: EOleException; Query: String);
26477: function UpdateBlob(Connection: TADOConnection; Spalte: String; Tabelle: String; Where: String; var ms: TMemoryStream): Boolean; ..save HTML pages as MHTML (HTML Archiv Format)
26478:   http://www.swissdelphicenter.ch/en/showcode.php?id=2300
26479: function SaveToMHT(const AUrl, AFileName: string; AShowErrorMessage: Boolean = False): Boolean;
26480: function FileType2MimeType(const AFileName: string): string;
26481: function DownloadURL_NOCache(const aUrl: string; var s: String): Boolean;
26482: Function IsCOMObjectActive(ClassName: string): Boolean;
26483: Procedure CopyHTMLOClipboard(const str: string; const htmlStr: string = '');
26484: function CheckCreditCard(c: string): Integer;
26485:   0: Card is invalid or unknown 1: is a AmEx 2: is a Visa 3: is a valid MasterCard
26486:   function Get.NewGuid: string;
26487:   function FormatGUID(const GUID: string): string;
26488:   function Get.NewFormatedGUID: string;
26489: function getFormRes(classname: string): string; //shows DFM Res of Exe!
26490: procedure OutputDebugString(PChar(Format('[%s][%s]%', [aCaption,GetFormatDT(StartDT),aText]));
26491: procedure ScanNetworkResources(ResourceType, DisplayType: DWord; List: TStrings);
26492:   function PrepareConstraint(Src:Tstrings):string;
26493: procedure DeleteEmptyStr(Src:Tstrings);
26494: function NormalizeSQLText(const SQL: string;MacroChar:Char): string;

```

```

26495:   function CountSelect (const SrcSQL:string) :string;
26496:   function GetModifyTable (const SQLText:string;AlreadyNormal:boolean) :string;
26497:   function GetCharFromVKey(vkey: Word): string;
26498:   function Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile: string): Boolean;
26499:   function IsObjectActive (ClassName: string): Boolean;
26500:   function GetActiveObject (ClassID:GUID; anil:TObject; aUnknown:IUnknown) :HRESULT;';
26501:   function RegisterOCX (FileName: string): Boolean;
26502:   function UnRegisterOCX (fileName: string): Boolean;
26503:   function RegisterServer2 (const aD11FileName: string; aRegister: Boolean): Boolean;
26504:   procedure mIRCDE (Service, Topic, Cmd: string); //mIRCDE('mIRC', 'COMMAND', '/say Hallo von
SwissDelphiCenter.ch');
26505:   function OpenIE (aURL: string): boolean;
26506:   function XRTLIsInMainThread: Boolean;
26507:   function IsInMainThread: Boolean;
26508:   TryConvertStrToDate (const s, format: string; out value: TDateTime): Boolean;'";
26509:   ConvertStrToDate (const s, format: string): TDateTime;'";
26510:   Function CreateNotifyThread2 (const FolderName : string; WatchSubtree : Boolean; Filter :
TFileChangeFilters2) : TNotifyThread;';
26511:   procedure DetectImage (const InputFileName: string; BM: TBitmap);
26512:   function BitmapToString (Bitmap: TBitmap): String;
26513:   function StringToBitmap (S: String): TBitmap;
26514:   FUNCTION RemoveChar (CONST s: STRING; CONST c: CHAR): STRING;
26515:   procedure SecureClearStr (var S: AnsiString);'";
26516:   procedure movestring (const Source:string; var Destination:string; CopyCount : Integer );
26517:   procedure moveint (const Source:integer; var destination: integer; CopyCount : Integer );
26518:   procedure movefloat (const Source:double; var destination: double; CopyCount : Integer );
26519:   procedure moveextended (const Source:extended;var Destination:extended;CopyCount:Int);
26520:   procedure ShowFilePropertiesSH (Files: TStrings; aWnd: HWND);
26521: //ScanNetworkResources (RESOURCETYPE_DISK, RESOURCEDISPLAYTYPE_SERVER, ListBox1.Items);
26522:   function GrabLine (const s: string; ALine: Integer): string;
26523:   function GrabLineFast (const s: string; ALine: Integer): string;
26524:   function IsTextFile (const sFile: TFileName): boolean;
26525:   function getODBC: TStringList;
26526:   function getODBCString: string;
26527:   procedure GetJPGSize (const sFile: string; var wWidth, wHeight: Word);
26528:   procedure GetPNGSize (const sFile: string; var wWidth, wHeight: Word);
26529:   procedure GetGIFSize (const sGIFFile: string; var wWidth, wHeight: Word);
26530: // note: using TForm's BorderIcons, etc. is slow (form blinks) and not reliable (for some
26531: // reson it causes TListView.Items to lose all associated objects and other things happen).
26532:   procedure ChangeWindowStyle (const Form: HWND; Style: DWord; AddIt: Boolean);
26533:
26534:   function ComputePEChecksum (FileName: string): DWord;
26535: if not DynamicD11CallName (user32, 'LockWorkStation', true, returned, parameters) then begin
26536:   function DynamicD11CallNames (Dll: String; const Name: String; HasResult: Boolean; var Returned: Cardinal;
const Parameters: array of string): Boolean;'";
26537:   Procedure GetMIDASAppServerList (List: TStringList; const RegCheck : string);';
26538:   procedure SQLdropField (dbName, tblName, fldName: String); {Field Name to Drop}
26539:   type TCastType = (ctSmallInt, ctInteger, ctDecimal, ctNumeric, ctFloat,
26540:                      ctChar, ctVarChar, ctDate, ctBoolean, ctBLOB, ctTime,
26541:                      ctTimeStamp, ctMoney, ctAutoInc, ctBytes);
26542: {Blob definition type 1 = Memo, 2 = Binary, 3 = Formatted Memo, 4 = OLE Object, 5 = Graphic}
26543:
26544:   procedure SQLAddField (dbName, tblName, fldName: String; fldType: TCastType; fldLength, precisOrBlobLen,
scaleOrBlobType: Integer);
26545:   const
26546:     UrlGeoLookupInfo = 'http://ipinfodb.com/ip_query.php?timezone=true&ip=%s';
26547:     UrlGeoLookupInfo2 = 'http://backup.ipinfodb.com/ip_query.php?timezone=true&ip=%s'; //backup
26548:
26549:   procedure GetGeoInfo (const IpAddress : string; var GeoInfo : TGeoInfo; const UrlGeoLookupInfo: string);
26550:   TGeoInfo', 'record status: string; countrycode : '
26551:     + string; countryname:string; regioncode: string; city: string; zippostalcode: string; latitude '
26552:     +: string; longitude:string; timezonename:string; gmtoffset: string; isdst:string; end');
26553:
26554:   procedure SIRegister_ubigFloatV3 (CL: TPSPascalCompiler);
26555:   begin
26556:     TMaxSig', 'integer';
26557:     TView', '( normal, Scientific )';
26558:     SIRegister_TFloatInt (CL); SIRegister_TBigFloat (CL);
26559:   end;
26560:
26561:   procedure SIRegister_UBigIntsV4 (CL: TPSPascalCompiler);
26562:   begin TDigits', 'array of int64'; SIRegister_TInteger(CL);
26563:   Procedure SetBaseVal ( const newbase : integer) ';
26564:   Function GetBasePower : integer)';
26565:   Function GetBase : integer)';
26566:   Procedure SetThreadSafe( newval : boolean)';
26567:   end;
26568:
26569:   function BigDiv(aone, atwo: string): string;
26570:   var tbig1, tbig2: TInteger;
26571:   begin
26572:     tbig1:= TInteger.create (10);
26573:     tbig2:= TInteger.create (10);
26574:     try
26575:       tbig1.assign2 (atwo)
26576:       tbig2.assign2 (aone)
26577:       tbig2.Divide (tbig1)
26578:     finally
26579:       result:= tbig2.Convert.ToDecimalString (false)

```

```

26580:     tbig1.Free;
26581:     tbig2.free;
26582:   end;
26583: end;
26584:
26585: procedure SIRegister_UDict(CL: TPSPPascalCompiler);
26586: begin 'dichighletter','String').SetString( 'z');
26587:   SIRegister_TDicForm(CL);
26588:   SIRegister_TDic(CL);
26589: end;
26590:
26591: procedure SetArrayLength2Char2(var arr: T2CharArray; asizel, asize2: integer);
26592: var i: integer;
26593: begin setlength(arr, asizel);
26594:   for i:= 0 to asizel do SetLength(arr[i], asize2);
26595: end;
26596:
26597: procedure SIRegister_UP10Build(CL: TPSPPascalCompiler);
26598: begin Procedure ParseFunction(FunctionString:string; Variables:TStringlist; FunctionOne,FunctionTwo: TStringList; UsePascalNumbers : boolean; var FirstTOP: (TObject)POperation; var Error : boolean)');
26599: end;
26600:
26601: procedure SIRegister_ComObj2(CL: TPSPPascalCompiler);
26602: begin
26603:   CL.AddClassN(CL.FindClass ('TOBJECT'), 'TComObjectFactory');
26604:   SIRegister_TComServerObject(CL);
26605:   SIRegister_ADOConst(CL);
26606:   FieldTypeNames: array[0..41] of string = (
26607:     'Unknown', 'String', 'SmallInt', 'Integer', 'Word', 'Boolean', 'Float',
26608:     'Currency', 'BCD', 'Date', 'Time', 'DateTime', 'Bytes', 'VarBytes',
26609:     'AutoInc', 'Blob', 'Memo', 'Graphic', 'FmtMemo', 'ParadoxOle',
26610:     'dBaseOle', 'TypedBinary', 'Cursor', 'FixedChar', 'WideString',
26611:     'LargeInt', 'ADT', 'Array', 'Reference', 'DataSet', 'HugeBlob', 'HugeClob',
26612:     'Variant', 'Interface', 'Dispatch', 'Guid', 'SQLTimeStamp', 'FMTBcdField',
26613:     'FixedWideChar', 'WideMemo', 'SQLTimeStamp', 'String');
26614:   TFactoryProc', 'Procedure (Factory : TComObjectFactory)';
26615:   TCallingConvention', '(ccRegister, ccDecl, ccPascal, ccStdCall, ccSafeCall)';
26616:   SIRegister_TComClassManager(CL);
26617:   SIRegister_IServerExceptionHandler(CL); SIRegister_TComObject(CL);
26618:   //TComClass', 'class of TComObject';
26619:   TClassInstancing', '(ciInternal, ciSingleInstance, ciMultiInstance )';
26620:   TThreadingModel', '(tmSingle, tmApartment, tmFree, tmBoth, tmNeutral )';
26621:   SIRegister_TComObjectFactory(CL); SIRegister_TTypedComObject(CL);
26622:   //TTypedComClass', 'class of TTypedComObject';
26623:   SIRegister_TTypedComObjectFactory(CL);
26624:   TConnectEvent2', 'Procedure (const Sink : IUnknown; Connecting : Boolean)';
26625:   CL.AddClassN(CL.FindClass ('TOBJECT'), 'TAutoObjectFactory');
26626:   SIRegister_TAutoObject(CL); //TAutoObject2 ?? in OleAuto and ComObj
26627:   //TAutoClass', 'class of TAutoObject';
26628:   SIRegister_TAutoObjectFactory(CL); SIRegister_TAutoIntfObject(CL);
26629:   //CL.AddClassN(CL.FindClass ('TOBJECT'), 'EOleError');
26630:   CL.AddClassN(CL.FindClass ('Exception'), 'EOleError');
26631:   SIRegister_EOLError(CL); SIRegister_EOLErrorException(CL);
26632:   CL.AddClassN(CL.FindClass ('TOBJECT'), 'EOleRegistrationError');
26633:   //Procedure DispatchInvoke( const Dispatch : IDispatch; CallDesc : PCallDesc; DispIDs : PDispIDList;
26634:   Params : Pointer; Result : PVariant)');
26635:   //Procedure DispatchInvokeError( Status : Integer; const ExcepInfo : TExcepInfo );
26636:   //Function HandleSafeCallException( ExceptObject : TObject; ExceptAddr : Pointer; const ErrorIID : TGUID;
26637:   const ProgID, HelpFileName : WideString ) : HResult );
26638:   Function CreateComObject( const ClassID : TGUID ) : IUnknown;
26639:   Function CreateRemoteComObject( const MachineName: WideString; const ClassID:TGUID):IUnknown;
26640:   //Function CreateOleObject(const ClassName: string): IDispatch;
26641:   //Function GetActiveOleObject(const ClassName:string): IDispatch;
26642:   Procedure OleError2( ErrorCode : HResult );
26643:   Procedure OleCheck2( Result : HResult );
26644:   Function StringToGUID2( const S : string ) : TGUID;
26645:   Function GUIDToString2( const ClassID : TGUID ) : string;
26646:   Function ProgIDToClassID2( const ProgID : string ) : TGUID;
26647:   Function ClassIDToProgID2( const ClassID : TGUID ) : string;
26648:   Procedure CreateRegKey( const Key, ValueName, Value:string; RootKey:DWord );
26649:   Procedure DeleteRegKey( const Key : string; RootKey : DWord );
26650:   Function GetRegStringValue( const Key, ValueName : string; RootKey : DWord ) : string;
26651:   //Function StringToLPOLESTR( const Source : string ) : PoleStr;
26652:   Procedure RegisterComServer( const DLLName : string );
26653:   Procedure RegisterAsService( const ClassID, ServiceName : string );
26654:   Function CreateClassID2 : string;
26655:   Procedure InterfaceConnect( const Source:IUnknown; const IID:TIID; const Sink:IUnknown; var Connection:Longint );
26656:   Function GetDispatchPropValue(Disp: IDispatch; DispID : Integer) : OleVariant;
26657:   Function GetDispatchPropValue1(Disp: IDispatch; Name : WideString) : OleVariant;
26658:   Procedure SetDispatchPropValue2(Disp: IDispatch; DispID:Integer; const Value:OleVariant);
26659:   Procedure SetDispatchPropValue3(Disp: IDispatch; Name:WideString; const Value:OleVariant);
26660:   // from ADODB OLE Utils
26661:   TOleEnum', 'LongWord'); //DataTypeEnum = TOleEnum;
26662:   DataTypeEnum', 'TOleEnum';
26663:   Function CreateADOObject( const ClassID : TGUID ) : IUnknown;
26664:   Function ADOTypeToFieldType( const ADOType:DataTypeEnum; EnableBCD:Boolean):TFieldType);

```

```

26665: Function FieldTypeToADOType(const FieldType: TFieldType): DataTypeEnum';
26666: Function StringToVarArray( const Value : string) : OleVariant';
26667: Function VarDataSize( const Value : OleVariant) : Integer';
26668: Function OleEnumToOrd( OleEnumArray : array of TOleEnum; Value : TOleEnum) : Integer';
26669: Function GetStates( State : Integer) : TObjectStates';
26670: Function ExecuteOptionsToOrd(ExecuteOptions:TExecuteOptions): Integer';
26671: Function OrdToExecuteOptions(Options: Integer) : TExecuteOptions';
26672: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString';
26673: Function GetFilterStr( Field : TField; Value : Variant; Partial : Boolean) : WideString';
26674: Function FieldListCheckSum( DataSet : TDataset) : Integer';
26675: function GlobalAllocString(s: AnsiString): HGlobal;
26676: function ScanTim( const S: string; var Pos: Integer; var Time: TDateTime): Boolean;
26677: function ScanChar( const S: string; var Pos: Integer; Ch: Char): Boolean;
26678: function ScanNumber( const S: string; var Pos: Integer; var Number: Word): Boolean;
26679: function ScanString( const S: string; var Pos: Integer; const Symbol: string): Boolean;
26680: procedure LV_InsertFiles( strPath: string; ListView: TListView; ImageList: TImageList);
26681: Function GetPasteLinkInfo( var Service: string; var Topic:string; var Item: string):Boolean';
26682: function IPToHostName( const IP: string): string;
26683: procedure GetZoneIcon(IconPath: string; var Icon: TIcon);
26684: function GetZoneAttributes( const URL: string): TZoneAttributes;
26685: //unit uPSI_PsAPI;
26686: procedure CGITester;
26687: //CGI will take name and email address from command line and place it into HTML
26688: procedure CreateBrowserOnForm(aform: TForm; aurl: string);
26689: procedure WebOnForm(aform: TForm; aurl: string);'
26690: procedure WebToForm(aform: TForm; aurl: string);'
26691: procedure SearchAndHighlightWebText(aform: TForm; aurl: string; aText: string);
26692: procedure SaveImagesOnWeb(aurl, apath: string);
26693: function GetProcessNameFromWnd(Wnd: HWND): string; //get EXE path from window handle
26694: function getallEvents(aform: TForm): TStringlist;
26695: procedure GetKList(List: TStrings);'
26696: procedure GetKeyboardList(List: TStrings);'
26697: function SetSuspendState(Hibernate:Boolean;ForceCritical:Boolean;DisableWakeEvent:Bool):bool;
26698: call: SetSuspendState(True, False, False);
26699: function ServiceRunning(sMachine, sService: PChar): Boolean;
26700: function isServiceRunning(sMachine, sService: PChar): Boolean;
26701: Procedure CloseOpenSockets( Sockets : array of TIdStackSocketHandle);
26702: //SIRregister_SvrHTTPIndy - Linux
26703: Procedure TransformError( const Msg : string)';
26704: Procedure StringToFile2( const S, FileName : string)';
26705: Function GetXMLData( DataSet : TClientDataSet) : string';
26706: Procedure EditComTerminal( ComTerminal : TCustomComTerminal); //TComTrmSetForm
26707: function search_adapter_key_networkcard: string; //at registry
26708: function getNetworkCard: string;'
26709: function GetMacAddresses2(const Machine: string; const Addresses: TStrings): Integer;
26710: function ConnectDrive(_drvLetter:string;_netPath:string;_showError:Boolean;_reconnect:Boolean):DWORD;
26711: function ConnectPrinterDevice(_lptPort:string;_netPath:string;_showError:Bool;_reconnect:Bool):DWORD;
26712: function DisconnectNetDrive(_locDrive: string;_showError:Bool;_force: Boolean; _save:Bool): DWORD;
26713: //ConnectDrive('k:', '\Servername\C', True, True);
26714: //DisconnectNetDrive('k:', True, True, True);
26715: function GetConnectionKind(var strKind: string): Boolean; MODEM =1; LAN=2; PROXY=4; BUSY= 8;
26716: function DownloadJPGToBitmap(const URL : string; ABitmap: TBitmap): Boolean;
26717: procedure GetImageLinks(AURL: string; AList: TStrings);
26718: Function GetCharEncoding( alias : string; var _name : string) : integer';
26719: Function MicrosoftCodePageToMIMECharset( cp : word) : string';
26720: Function MicrosoftLangageCodeToISOCode( langcode : integer) : string';
26721: procedure CopyHTMLToClipboard(const str: string; const htmlStr: string = '');
26722: function RFC1123ToDateTime(Date: string): TDateTime;
26723: function DateToStringRFC1123(aDate: TDateTime): string;
26724: procedure CopyHTMLToClipboard(const str: string; const htmlStr: string);'
26725: procedure DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);'
26726: procedure DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);'
26727: procedure DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);'
26728:
26729:
26730: IPToHostName
26731: //from ADOInt.pas // TOleEnum = type LongWord;
26732: CL.AddTypeS('CursorOptionEnum', 'TOleEnum');
26733: CL.AddInterface(CL.FindInterface('IUNKNOWN'), _Recordset, '_Recordset');
26734: //CL.AddTypeS('_RecordsetDisp', 'dispinterface');
26735: CL.AddInterface(CL.FindInterface('IUNKNOWN'), _Command, '_Command');
26736: CL.AddInterface(CL.FindInterface('IUNKNOWN'), _Connection, '_Connection');
26737: // SIRregister_Recordset(CL); // SIRregister_Command(CL);
26738: Const IID_Recordset20': TGUID').SetString('{0000054F-0000-0010-8000-00AA006D2EA4}''']);
26739: 'IID_Recordset','String').SetString( '00000555-0000-0010-8000-00AA006D2EA4');
26740: //test with stringTOGUID
26741: //CLASS_Command', 'TGUID').SetString( '00000507-0000-0010-8000-00AA006D2EA4');
26742: // 'CLASS_Recordset', 'TGUID').SetString( '{00000535-0000-0010-8000-00AA006D2EA4}');
26743: { CL.AddTypeS('Connection', '_Connection');
26744: CL.AddTypeS('Command', '_Command'); CL.AddTypeS('Recordset', '_Recordset');
26745: CL.AddTypeS('Parameter', '_Parameter'); CL.AddTypeS('DataSpace', 'IDataspace');
26746: CL.AddTypeS('SearchDirection', 'SearchDirectionEnum'); }
26747: //CL.AddTypeS('Command', '_Command'); //CL.AddTypeS('Recordset', '_Recordset');
26748: end;
26749:
26750: procedure SIRregister_DIUtils(CL: TPPascalCompiler);
26751: begin
26752: //'\r\n', 'String').SetString( #$0D#$0A);
26753: 'REPLACEMENT_CHARACTER', 'LongWord').SetUInt( $FFFFD);

```

```

26754: 'HANGUL_SBase','LongWord').SetUInt( $AC00);
26755: 'HANGUL_LBase','LongWord').SetUInt( $1100);
26756: 'HANGUL_VBase','LongWord').SetUInt( $1161);
26757: 'HANGUL_TBase','LongWord').SetUInt( $11A7);
26758: 'HANGUL_LCount','LongInt').SetInt( 19);
26759: 'HANGUL_VCount','LongInt').SetInt( 21);
26760: 'HANGUL_TCount','LongInt').SetInt( 28);
26761: 'KEY_WOW64_32KEY','LongWord').SetUInt( $0200);
26762: 'KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
26763: 'KEY_WOW64_RES','LongWord').SetUInt( $0300);
26764: CL.AddTypeS('TAnsiCharSet', 'set of AnsiChar');
26765: CL.AddTypeS('TIsoDate', 'Cardinal');
26766: CL.AddTypeS('TJulianDate', 'Integer');
26767: //CL.AddTypeS('TPJulianDate', '^TJulianDate // will not work');
26768: CL.AddTypeS('TValidateCharFunc', 'function(const c: Char): Boolean;');
26769: // TValidateCharFunc = function( const c: Char): Boolean;
26770: CL.AddTypeS('TProcedureEvent', 'Procedure');
26771: 'MT19937_N','LongInt').SetInt( 624);
26772: 'MT19937_M','LongInt').SetInt( 397);
26773: SIRegister_TMT19937(CL);
26774: //SIRegister_TWideStrBuf(CL);
26775: CL.AddTypeS('TDITextLineBreakStyle', '( tlbsLF, tlbsCRLF, tlbsCR )');
26776: Function AdjustLineBreaksW(const s: UnicodeString; const Style:TDITextLineBreakStyle):UnicodeString;
26777: Function BrightenColor( const Color : Integer; const amount : Byte) : Integer';
26778: Function BSwap4( const Value : Cardinal) : Cardinal;';
26779: Function BSwap5( const Value : Integer) : Integer';
26780: //Function BufCompNumIW(p1:PWideChar; l1:Integer; p2 : PWideChar; 12 : Integer) : Integer';
26781: Function BufSameA( p1, p2 : PChar; l : Cardinal) : Boolean';
26782: // Function BufSameW( p1, p2 : PWideChar; l : Cardinal) : Boolean';
26783: Function BufSameIA( p1, p2 : PChar; l : Cardinal) : Boolean';
26784: //Function BufSameIW( p1, p2 : PWideChar; l : Cardinal) : Boolean';
26785: Function BufPosCharA(const Buf:PChar;l:Cardinal;const c:AnsiChar,const Start:Cardinal:Int');
26786: Function BufPosCharsA(const Buf:PChar;l:Card;const Search:TAnsiCharSet;const Start:Card):Int;
26787: Function BufStrSame(const Buf:PChar;const BufCharCount:Cardinal;const s: string): Boolean';
26788: Function BufStrSameA(const Buf:PChar; const BufCharCount:Cardinal;const s:RawByteString):Bool;
26789: Function BufStrSameI(const Buf:PChar; const BufCharCount:Cardinal; const s : string): Boolean;
26790: Function BufStrSameIA(const Buf:PChar;const BufCharCount:Cardinal;const s:RawByteString): Bool;
26791: Function diChangeFileExt(const FileName,Extension: string): string';
26792: Function ChangeFileExtA( const FileName, Extension : AnsiString) : AnsiString';
26793: Function ChangeFileExtW( const FileName, Extension : UnicodeString) : UnicodeString';
26794: //Function CharDecomposeCanonicalW( const c : WideChar) : PCharDecompositionW';
26795: Function CharDecomposeCanonicalStrW( const c : WideChar) : UnicodeString';
26796: //Function CharDecomposeCompatibleW( const c : WideChar) : PCharDecompositionW';
26797: Function CharDecomposeCompatibleStrW( const c : WideChar) : UnicodeString';
26798: Function CharIn8( const c, t1, t2 : WideChar) : Boolean';
26799: Function CharIn9( const c, t1, t2, t3 : WideChar) : Boolean';
26800: Procedure ConCatBuf(const Buffer: PChar;const CharCount: Cardinal;var d:string;var InUse:Card);
26801: Procedure ConCatBufA(const Buffer:PChar;const AnsiCharCount:Card;var d:RawByteString;var InUse:Card);
26802: Procedure ConCatChar( const c : Char; var d : string; var InUse : Cardinal)';
26803: Procedure ConCatCharA( const c : AnsiChar; var d : RawByteString; var InUse : Cardinal)');
26804: // Procedure ConCatCharW( const c : WideChar; var d : UnicodeString; var InUse : Cardinal)');
26805: Procedure ConCatStr( const s : string; var d : string; var InUse : Cardinal)');
26806: Procedure ConCatStrA( const s: RawByteString; var d : RawByteString; var InUse : Cardinal)');
26807: //Procedure ConCatStrW(const w:UnicodeString; var d : UnicodeString; var InUse : Cardinal)');
26808: Function diCountBitsSet( const Value : Integer) : Byte';
26809: //Function Crc32OfStrA( const s : RawByteString) : TCrc32';
26810: //Function Crc32OfStrW( const s : UnicodeString) : TCrc32';
26811: Function CurrentDay : Word';
26812: Function CurrentJulianDate : TJulianDate';
26813: Function CurrentMonth : Word');
26814: Function CurrentQuarter : Word');
26815: Function diCurrentYear : Integer';
26816: Function DarkenColor( const Color : Integer; const amount : Byte) : Integer';
26817: Function diDeletefile( const FileName : string) : Boolean';
26818: Function DeleteFileA( const FileName : AnsiString) : Boolean';
26819: //Function DeleteFileW( const FileName : UnicodeString) : Boolean';
26820: Function diDirectoryExists( const Dir : string) : Boolean';
26821: Function DirectoryExistsA( const Dir : AnsiString) : Boolean';
26822: //Function DirectoryExistsW( const Dir : UnicodeString) : Boolean';
26823: Function diDiskFree( const Dir : string) : Int64';
26824: Function DiskFreeA( const Dir : AnsiString) : Int64';
26825: //Function DiskFreeW( const Dir : UnicodeString) : Int64';
26826: Function diExpandFileName( const FileName : string) : string';
26827: Function ExpandFileNameA(const FileName: AnsiString): Ansistring';
26828: //Function ExpandFileNameW( const FileName : UnicodeString) : UnicodeString';
26829: Procedure diExcludeTrailingPathDelimiter( var s : string)';
26830: Procedure ExcludeTrailingPathDelimiterA( var s : RawByteString)';
26831: //Procedure ExcludeTrailingPathDelimiterW( var s : UnicodeString)';
26832: Function diExtractFileDrive( const FileName : string) : string';
26833: Function ExtractFileDriveA( const FileName : RawByteString) : RawByteString';
26834: //Function ExtractFileDriveW( const FileName : UnicodeString) : UnicodeString';
26835: Function diExtractFileExt( const FileName : string) : string';
26836: Function ExtractFileExtA( const FileName : RawByteString) : RawByteString';
26837: //Function ExtractFileExtW( const FileName : UnicodeString) : UnicodeString';
26838: Function diExtractFileName( const FileName : string) : string';
26839: Function ExtractFileNameA(const FileName: AnsiString): Ansistring';
26840: //Function ExtractFileNameW( const FileName : UnicodeString) : UnicodeString';
26841: Function diExtractFilePath( const FileName : string) : string';
26842: Function ExtractFilePathA( const FileName : RawByteString) : RawByteString';

```

```

26843: //Function ExtractFilePathW( const FileName : UnicodeString ) : UnicodeString';
26844: Function ExtractNextWord10(const s:string;const ADelimiter:Char;var AstartIndex:Int):string';
26845: Function ExtractNextWordA11(const s: RawByteString; const ADelimiter : AnsiChar; var AstartIndex : Integer) : RawByteString;';
26846: Function ExtractNextWordW12(const s : UnicodeString; const ADelimiter:WideChar; var AstartIndex : Integer) : UnicodeString;';
26847: Function ExtractNextWord13(const s:string;const ADelims:TAnsiCharSet;var AstartIndex:Integer):string';
26848: Function ExtractNextWordA14(const s:RawByteString; const ADelimiters:TAnsiCharSet; var AstartIndex : Integer) : RawByteString;';
26849: Function diExtractWord( const Number : Cardinal; const s : RawByteString; const Delimiters : TAnsiCharSet ) : RawByteString;';
26850: Function ExtractWordA( const Number : Cardinal; const s : RawByteString; const Delimiters : TAnsiCharSet ) : RawByteString';
26851: Function ExtractWordStartsA( const s : RawByteString; const MaxCharCount : Cardinal; const WordSeparators : TAnsiCharSet ) : RawByteString';
26852: Function diFileExists( const FileName : string ) : Boolean';
26853: Function FileExistsA( const FileName : AnsiString ) : Boolean';
26854: //Function FileExistsW( const FileName : UnicodeString ) : Boolean';
26855: Function diGCD( x, y : Cardinal ) : Cardinal';
26856: Function diGetTempFolder : string';
26857: Function GetTempFolderA : AnsiString';
26858: //Function GetTempFolderW : UnicodeString';
26859: Function diGetUserName( out UserName : string ) : Boolean';
26860: Function GetUserNameA( out UserName : AnsiString ) : Boolean';
26861: //Function GetUserNameW( out UserName : UnicodeString ) : Boolean';
26862: Function HexCodePointToInt( const c : Cardinal ) : Integer';
26863: Function diHexToInt( const s : string ) : Integer';
26864: Function HexToIntA( const s : RawByteString ) : Integer';
26865: //Function HexToIntW( const s : UnicodeString ) : Integer';
26866: Function BufHexToInt( p : PChar; l : Cardinal ) : Integer';
26867: Function BufHexToIntA( p : PChar; l : Cardinal ) : Integer';
26868: //Function BufHexToIntW( p : PWideChar; l : Cardinal ) : Integer';
26869: Procedure IncludeTrailingPathDelimiterByRef( var s : string );
26870: Procedure IncludeTrailingPathDelimiterByRefA(var s:RawByteString');
26871: //Procedure IncludeTrailingPathDelimiterByRefW(var w:UnicodeString)';
26872: Function IntToHex16( const Value : Integer; const Digits : NativeInt ) : string';
26873: Function IntToHex17( const Value : Int64; const Digits : NativeInt ) : string';
26874: Function IntToHex18( const Value : UInt64; const Digits : NativeInt ) : string';
26875: Function IntToHexA( Value : UInt64; const Digits : NativeInt ) : RawByteString';
26876: //Function IntToHexW( Value : UInt64; const Digits : NativeInt ) : UnicodeString';
26877: Function IntToStrA19( const i : Integer ) : RawByteString';
26878: Function IntToStrW20( const i : Integer ) : UnicodeString';
26879: Function IntToStrA21( const i : Int64 ) : RawByteString';
26880: Function IntToStrW22( const i : Int64 ) : UnicodeString';
26881: Function CharDecomposeHangulW(const c: WideChar) : UnicodeString';
26882: Function diIsPathDelimiter( const s : string; const Index : Cardinal ) : Boolean';
26883: Function IsPathDelimiterA( const s : RawByteString; const Index : Cardinal ) : Boolean';
26884: //Function IsPathDelimiterW( const s : UnicodeString; const Index : Cardinal ) : Boolean';
26885: Function IsPointInRect( const Point : TPoint; const Rect : TRect ) : Boolean';
26886: Function JulianDateToIsoDateStr(const Julian: TJJulianDate):string';
26887: Function JulianDateToIsoDateStrA( const Julian : TJJulianDate ) : RawByteString';
26888: //Function JulianDateToIsoDateStrW( const Julian : TJJulianDate ) : UnicodeString';
26889: Function LeftMostBit( Value : Cardinal ) : ShortInt';
26890: Function LeftMostBit2( Value : UInt64 ) : ShortInt';
26891: //Function MakeMethod( const AData, ACode : Pointer ) : TMethod';
26892: Function StrIsEmpty( const s : string ) : Boolean';
26893: Function StrIsEmptyA( const s : RawByteString ) : Boolean';
26894: //Function StrIsEmptyW( const s : UnicodeString ) : Boolean';
26895: Function PadLeftA(const Source:RawByteString;const Count:Cardinal; const c:AnsiChar):RawByteString;
26896: //Function PadLeftW(const Source : UnicodeString; const Count : Cardinal; const c : WideChar) : UnicodeString';
26897: Function PadRightA(const Source: RawByteString; const Count:Cardinal;const c:AnsiChar):RawByteString;
26898: //Function PadRightW(const Source : UnicodeString; const Count : Cardinal; const c : WideChar) : UnicodeString';
26899: Function ProperCase( const s : string ) : string';
26900: Function ProperCaseA( const s : RawByteString ) : RawByteString';
26901: //Function ProperCaseW( const s : UnicodeString ) : UnicodeString';
26902: Procedure ProperCaseByRefA( var s : RawByteString );
26903: //Procedure ProperCaseByRefW( var s : UnicodeString );
26904: Function RegReadRegisteredOrganization( const Access : REGSAM ) : string';
26905: Function RegReadRegisteredOrganizationA( const Access : REGSAM ) : AnsiString';
26906: //Function RegReadRegisteredOrganizationW( const Access : REGSAM ) : UnicodeString';
26907: Function RegReadRegisteredOwner( const Access : REGSAM ) : string';
26908: Function RegReadRegisteredOwnerA( const Access : REGSAM ) : AnsiString';
26909: //Function RegReadRegisteredOwnerW( const Access : REGSAM ) : UnicodeString';
26910: Function RegReadStrDef( const Key : HKEY; const SubKey : string; const ValueName : string; const Default : string; const Access : REGSAM ) : string';
26911: Function RegReadStrDefA( const Key : HKEY; const SubKey : AnsiString; const ValueName : AnsiString; const Default : AnsiString; const Access : REGSAM ) : AnsiString';
26912: Function StrDecodeUrlA( const Value : RawByteString ) : RawByteString';
26913: Function StrEncodeUrlA( const Value : RawByteString ) : RawByteString';
26914: Function diStrEnd( const s : PChar ) : PChar';
26915: Function StrEndA( const s : PChar ) : PChar';
26916: //Function StrEndW( const s : PWideChar ) : PWideChar';
26917: Procedure StrIncludeTrailingChar( var s : string; const c : Char );
26918: Procedure StrIncludeTrailingCharA( var s : RawByteString; const c : AnsiChar );
26919: //Procedure StrIncludeTrailingCharW( var s : UnicodeString; const c : WideChar );
26920: Function diStrLen( const s : PChar ) : NativeUInt';
26921: Function StrLenA( const s : PChar ) : NativeUInt';

```

```

26922: //Function StrLenW( const s : PWideChar ) : NativeUInt';
26923: Function StrRandom(const ASeed:RawByteStr;const ACharacters:string;const ALengt:Card):string';
26924: Function StrRandomA( const ASeed : RawByteString; const ACharacters : RawByteString; const ALengt :
Cardinal ) : RawByteString');
26925: Procedure StrRemoveFromToIA(var Source:RawByteString;const FromString,ToString:RawByteString)');
26926: //Procedure StrRemoveFromToIW(var Source:UnicodeString;const FromString, ToString : UnicodeString ');
26927: Procedure StrRemoveSpacingA(var s:RawByteString;const SpaceChars:TAnsiCharSet;const ReplaceChar:AnsiChar);
26928: Procedure diStrReplaceChar( var Source : string; const SearchChar, ReplaceChar : Char)');
26929: Procedure StrReplaceChar8( var s : Utf8String; const SearchChar, ReplaceChar : AnsiChar)');
26930: Procedure StrReplaceCharA(var s : RawByteString; const SearchChar, ReplaceChar : AnsiChar)');
26931: //Procedure StrReplaceCharW(var s:UnicodeString; const SearchChar, ReplaceChar : WideChar)');
26932: Function diStrReplace( const Source, Search, Replace : string ) : string');
26933: Function StrReplaceA( const Source, Search, Replace : RawByteString ) : RawByteString');
26934: //Function StrReplaceW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
26935: Function StrReplaceI( const Source, Search, Replace : string ) : string');
26936: Function StrReplaceIA( const Source, Search, Replace : RawByteString ) : RawByteString');
26937: //Function StrReplaceIW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
26938: Function StrReplaceLoopA( const Source, Search, Replace : RawByteString ) : RawByteString');
26939: //7Function StrReplaceLoopW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
26940: Function StrReplaceLoopIA( const Source, Search, Replace : RawByteString ) : RawByteString');
26941: //Function StrReplaceLoopIW( const Source, Search, Replace : UnicodeString ) : UnicodeString );
26942: Function RightMostBit( const Value : Cardinal ) : ShortInt');
26943: Function RightMostBit2( const Value : UInt64 ) : ShortInt');
26944: Function LoadStrFromFile( const FileName : string; var s : RawByteString ) : Boolean');
26945: Function FileToStr( const FileName : string; var s : String ) : Boolean');
26946: Function LoadStrAFFromFileA( const FileName : AnsiString; var s : RawByteString ) : Boolean');
26947: //Function LoadStrAFFromFileW(const FileName:UnicodeString; var s: RawByteString) : Boolean';
26948: Function LoadStrWFromFile28( const FileName : string; var s : UnicodeString ) : Boolean');
26949: Function LoadStrWFromFileA( const FileName : Ansistring; var s : UnicodeString ) : Boolean');
26950: //Function LoadStrWFromFileW( const FileName:UnicodeString; var s:UnicodeString) : Boolean');
26951: Function QuotedStrW( const s : UnicodeString; const Quote : WideChar ) : UnicodeString');
26952: Function SaveStrToFile( const s : string; const FileName : string ) : Boolean');
26953: Function StrToFile( const s : string; const FileName : string ) : Boolean');
26954: Function SaveStrAToFile( const s : RawByteString; const FileName : string ) : Boolean');
26955: Function SaveStrAToFileA( const s : RawByteString; const FileName : AnsiString ) : Boolean');
26956: //Function SaveStrAToFileW(const s:RawByteString; const FileName: UnicodeString) : Boolean');
26957: Function SaveStrWToFile( const s : UnicodeString; const FileName : string ) : Boolean');
26958: Function SaveStrWToFileA( const s : UnicodeString; const FileName : AnsiString ) : Boolean');
26959: //Function SaveStrWToFileW(const s:UnicodeString; const FileName : UnicodeString): Boolean');
26960: Function StrPosChar( const Source : string; const c:Char; const Start : Cardinal ) : Cardinal');
26961: Function StrPosCharA( const Source:RawByteString;const c:AnsiChar;const Start:Cardinal):Cardinal');
26962: //Function StrPosCharW(const Source:UnicodeString; const c:WideChar; const Start:Cardinal):Cardinal;
26963: Function StrPosCharBack( const Source : string; const c:Char; const Start:Cardinal ) : Cardinal');
26964: Function StrPosCharBackA( const Source:RawByteString;const c:AnsiChar;const Start:Cardinal):Cardinal);
26965: Function StrPosCharsA( const Source:RawByteString;const Search:TAnsiCharSet;const Start:Cardinal):
Cardinal;
26966: Function StrPosCharsBackA( const Source:RawByteString;const Search:TAnsiCharSet;const Start:Cardinal):
Cardinal;
26967: Function StrPosNotCharsA( const Source:RawByteString;const Search:TAnsiCharSet;const Start:Cardinal):
Cardinal;
26968: Function StrPosNotCharsBackA( const Source:RawByteString;const Search:TAnsiCharSet; const Start: Cardinal)
: Cardinal');
26969: Function SetFileDate( const FileHandle:THandle;const Year:Integer;const Month,Day:Word):Bool);
26970: Function SetFileDate2( const FileName : string; const JulianDate : TJ JulianDate ) : Boolean');
26971: Function SetFileDateA( const FileName : AnsiString; const JulianDate:T JulianDate) : Boolean');
26972: //Function SetFileDateW( const FileName : UnicodeString; const JulianDate : TJ JulianDate ) : Boolean');
26973: Function SetFileDateYmd( const FileName:string;const Year:Integer;const Month,Day:Word):Bool;
26974: Function SetFileDateYmdA( const FileName:Ansistring;const Year:Int;const Month,Day:Word):Bool;
26975: Function SetFileDateYmdW( const FileName:UnicodeString;const Year:Integer;const Month,Day:Word): Bool;
26976: Function StrContainsChar( const s:string; const c:Char; const Start : Cardinal ) : Boolean');
26977: Function StrContainsCharA( const s:RawByteString;const c:AnsiChar;const Start:Cardinal):Bool;
26978: //Function StrContainsCharW(const s : UnicodeString;const c:WideChar;const Start:Cardinal):Boolean';
26979: Function StrContainsCharsA( const s:RawByteStr;const Chars:TAnsiCharSet;const Start:Card):Bool;
26980: Function diStrSame( const s1, s2 : string ) : Boolean');
26981: Function StrSameA( const s1, s2 : RawByteString ) : Boolean');
26982: //Function StrSameW( const s1, s2 : UnicodeString ) : Boolean');
26983: Function StrSameI( const s1, s2 : string ) : Boolean');
26984: Function StrSameIA( const s1, s2 : RawByteString ) : Boolean');
26985: //Function StrSameIW( const s1, s2 : UnicodeString ) : Boolean');
26986: Function StrSameStart( const s1, s2 : string ) : Boolean');
26987: Function StrSameStartA( const s1, s2 : RawByteString ) : Boolean');
26988: //Function StrSameStartW( const s1, s2 : UnicodeString ) : Boolean');
26989: Function StrSameStartI( const s1, s2 : string ) : Boolean');
26990: Function StrSameStartIA( const s1, s2 : RawByteString ) : Boolean');
26991: //Function StrSameStartIW( const s1, s2 : UnicodeString ) : Boolean');
26992: Function diStrComp( const s1, s2 : string ) : Integer');
26993: Function StrCompA( const s1, s2 : RawByteString ) : Integer');
26994: //Function StrCompW( const s1, s2 : UnicodeString ) : Integer');
26995: Function StrCompI( const s1, s2 : string ) : Integer');
26996: Function StrCompIA( const s1, s2 : RawByteString ) : Integer');
26997: //Function StrCompIW( const s1, s2 : UnicodeString ) : Integer');
26998: Function StrCompNum( const s1, s2 : string ) : Integer');
26999: Function StrCompNumA( const s1, s2 : RawByteString ) : Integer');
27000: // Function StrCompNumW( const s1, s2 : UnicodeString ) : Integer');
27001: Function StrCompNumI( const s1, s2 : string ) : Integer');
27002: Function StrCompNumIA( const s1, s2 : RawByteString ) : Integer');
27003: // Function StrCompNumIW( const s1, s2 : UnicodeString ) : Integer');
27004: Function StrContains( const Search, Source : string; const Start : Cardinal ) : Boolean');
27005: Function StrContainsA( const Search, Source : RawByteString; const Start: Cardinal): Boolean');

```

```

27006: //Function StrContainsW(const ASearch, ASource: UnicodeString;const AStartPos: Cardinal): Boolean';
27007: Function StrContainsI( const Search, Source : string; const Start : Cardinal) : Boolean';
27008: Function StrContainsIA(const Search,Source:RawByteString; const Start: Cardinal): Boolean';
27009: //Function StrContainsIW(const ASearch,ASource:UnicodeString;const AStartPos:Cardinal):Bool;
27010: Function StrCountChar(const ASource:string;const c:Char;const AStartIdx:Cardinal):Cardinal';
27011: Function StrCountCharA(const ASource:RawByteString;const c:AnsiChar;const AStartIdx:Cardinal): Card;
27012: Function StrMatchesA(const Search,Source:RawByteString; const AStartIdx: Cardinal): Boolean';
27013: Function StrMatchesIA( const Search, Source : RawByteString; const AStartIdx : Cardinal) : Boolean';
27014: Function StrMatchWild(const Source,Mask:string;const WildChar:Char;const MaskChar:Char):Bool;
27015: Function StrMatchWildA(const Src,Mask:RawByteString;const WildChr:AnsiChar;const MaskChr:AnsiChar):Bool;
27016: Function StrMatchWildI(const Source,Mask:string;const WildChar:Char;const MaskChar:Char):Bool;
27017: Function StrMatchWildIA(const Source,Mask:RawByteString;const WildChar:AnsiChar;const
27018: MaskChar:AnsiChar):Bool;
27019: Function diStrPos( const ASearch, ASource : string; const AStartPos : Cardinal) : Cardinal';
27020: Function StrPosA( const ASearch,ASource: RawByteString; const AStartPos:Cardinal):Cardinal';
27021: //Function StrPosW( const ASearch, ASouRce : UnicodeString; const AStartPos : Cardinal) : Cardinal';
27022: Function StrPosI( const ASearch, ASource : string; const AStartPos : Cardinal) : Cardinal';
27023: //Function StrPosIW( const ASearch, ASource : UnicodeString; const AStartPos : Cardinal) : Cardinal';
27024: Function StrPosBackA( const ASearch, ASource: RawByteString; AStart : Cardinal) : Cardinal';
27025: Function StrPosBackIA(const ASearch, ASource: RawByteString; AStart : Cardinal) : Cardinal';
27026: //Function StrToIntDefW( const w : UnicodeString; const Default : Integer) : Integer';
27027: Function StrToInt64DefW( const w : UnicodeString; const Default : Int64) : Int64';
27028: Function StrToUpper( const s : string ) : string';
27029: Function StrToUpperA( const s : RawByteString ) : RawByteString';
27030: //Function StrToUpperW( const s : UnicodeString ) : UnicodeString';
27031: Procedure StrToUpperInPlace( var s : string );
27032: Procedure StrToUpperInPlaceA( var s : AnsiString );
27033: //Procedure StrToUpperInPlaceW31( var s : WideString );';
27034: //Procedure StrToUpperInPlaceW32( var s : UnicodeString );';
27035: Function StrToLower( const s : string ) : string';
27036: Function StrToLowerA( const s : RawByteString ) : RawByteString';
27037: //Function StrToLowerW( const s : UnicodeString ) : UnicodeString';
27038: Procedure StrToLowerInPlace( var s : string );
27039: Procedure StrToLowerInPlaceA( var s : AnsiString );
27040: //Procedure StrToLowerInPlaceW33( var s : WideString );';
27041: //Procedure StrToLowerInPlaceW34( var s : UnicodeString );';
27042: Procedure StrTimUriFragmentA( var Value : RawByteString );
27043: //Procedure StrTrimUriFragmentW( var Value : UnicodeString );
27044: //Function StrExtractUriFragmentW( var Value : UnicodeString ) : UnicodeString';
27045: Function StrCountUtf8Chars(const AValue: Utf8String) : Cardinal';
27046: Function StrDecodeUtf8( const AValue: Utf8String): UnicodeString';
27047: Function StrEncodeUtf8( const AValue: UnicodeString): Utf8String';
27048: Function diSysErrorMessage( const MessageID : Cardinal) : string';
27049: Function SysErrorMessageA(const MessageID: Cardinal): AnsiString';
27050: //Function SysErrorMessageW( const MessageID : Cardinal) : UnicodeString';
27051: Function TextExtentW( const DC : HDC; const Text : UnicodeString) : TSize';
27052: Function TextHeightW( const DC : HDC; const Text : UnicodeString) : Integer';
27053: Function TextWidthW( const DC : HDC; const Text : UnicodeString) : Integer';
27054: Function diStrTrim( const Source : string ) : string';
27055: Function StrTrimA( const Source : RawByteString ) : RawByteString';
27056: //Function StrTrimW( const w : UnicodeString ) : UnicodeString';
27057: Function StrTrimCharA(const Source:RawByteString;const CharToTrim:AnsiChar): RawByteString';
27058: Function StrTrimCharsA(const Source:RawByteString;const CharsToTrim:TAnsiCharSet):RawByteString;
27059: //Function StrTrimCharsW(const s:UnicodeString; const IsCharToTrim:TValidateCharFuncW):UnicodeString;
27060: Procedure TrimLeftByRefA( var s : RawByteString; const Chars : TAnsiCharSet );
27061: Function TrimRightA( const Source : RawByteString; const s : TAnsiCharSet ) : RawByteString';
27062: Procedure TrimRightByRefA( var Source : RawByteString; const s : TAnsiCharSet );
27063: Procedure StrTrimCompressA(var s:RawByteString;const TrimCompressChars:TAnsiCharSet;const
ReplaceChar:AnsiChar);
27064: Function TryStrToIntW( const w : UnicodeString; out Value : Integer ) : Boolean';
27065: Function TryStrToInt64W( const w : UnicodeString; out Value : Int64 ) : Boolean';
27066: Function ValInt( const p : PChar; const l : Integer; out Code : Integer ) : Integer; );
27067: Function ValIntA36( p : PChar; l : Integer; out Code : Integer ) : Integer; );
27068: //Function ValIntW37( p : PWideChar; l : Integer; out Code : Integer ) : Integer; );
27069: Function ValInt2(const s: string; out Code : Integer): Integer; );
27070: Function ValIntA39( const s : RawByteString; out Code : Integer ) : Integer; );
27071: Function ValIntW40( const s : UnicodeString; out Code : Integer ) : Integer; );
27072: Function ValInt64A41( p : PChar; l : Integer; out Code : Integer ) : Int64; );
27073: //Function ValInt64W42( p : PWideChar; l : Integer; out Code : Integer ) : Int64; );
27074: Function ValInt64A43( const s : RawByteString; out Code : Integer ) : Int64; );
27075: Function ValInt64W44( const s : UnicodeString; out Code : Integer ) : Int64; );
27076: Function YmdIsoDateStr(const Year:Integer; const Month: Word; const Day : Word): string';
27077: Function YmdIsoDateStrA(const Year:Int;const Month:Word; const Day:Word):RawByteString';
27078: Function YmdIsoDateStrW(const Year:Int;const Month:Word; const Day:Word):UnicodeString';
27079: Function CharIsLetterW( const c : WideChar ) : Boolean';
27080: Function CharIsLetterCommonW( const c : WideChar ) : Boolean';
27081: Function CharIsLetterUpperCaseW( const c : WideChar ) : Boolean';
27082: Function CharIsLetterLowerCaseW( const c : WideChar ) : Boolean';
27083: Function CharIsLetterTitleCaseW( const c : WideChar ) : Boolean';
27084: Function CharIsLetterModifierW( const c : WideChar ) : Boolean';
27085: Function CharIsLetterOtherW( const c : WideChar ) : Boolean';
27086: Function CharIsMarkW( const c : WideChar ) : Boolean';
27087: Function CharIsMarkNon_SpacingW( const c : WideChar ) : Boolean';
27088: Function CharIsMarkSpacing_CombinedW(const c : WideChar): Boolean';
27089: Function CharIsMarkEnclosingW( const c : WideChar ) : Boolean';
27090: Function CharIsNumberW( const c : WideChar ) : Boolean';
27091: Function CharIsNumber_DecimalW( const c : WideChar ) : Boolean';
27092: Function CharIsNumber_LetterW( const c : WideChar ) : Boolean');

```

```

27093: Function CharIsNumber_OtherW( const c : WideChar) : Boolean';
27094: Function CharIsPunctuationW( const c : WideChar) : Boolean';
27095: Function CharIsPunctuation_ConnectorW( const c : WideChar) : Boolean';
27096: Function CharIsPunctuation_DashW( const c : WideChar) : Boolean';
27097: Function CharIsPunctuation_OpenW( const c : WideChar) : Boolean';
27098: Function CharIsPunctuation_CloseW( const c : WideChar) : Boolean';
27099: Function CharIsPunctuation_InitialQuoteW( const c : WideChar) : Boolean';
27100: Function CharIsPunctuation_FinalQuoteW( const c : WideChar) : Boolean';
27101: Function CharIsPunctuation_OtherW( const c : WideChar) : Boolean';
27102: Function CharIsSymbolW( const c : WideChar) : Boolean';
27103: Function CharIsSymbolMathW( const c : WideChar) : Boolean';
27104: Function CharIsSymbolCurrencyW( const c : WideChar) : Boolean';
27105: Function CharIsSymbolModifierW( const c : WideChar) : Boolean';
27106: Function CharIsSymbolOtherW( const c : WideChar) : Boolean';
27107: Function CharIsSeparatorW( const c : WideChar) : Boolean';
27108: Function CharIsSeparatorSpaceW( const c : WideChar) : Boolean';
27109: Function CharIsSeparatorLineW( const c : WideChar) : Boolean';
27110: Function CharIsSeparatorParagraphW( const c : WideChar) : Boolean';
27111: Function CharIsOtherW( const c : WideChar) : Boolean';
27112: Function CharIsOtherControlW( const c : WideChar) : Boolean';
27113: Function CharIsOtherFormatW( const c : WideChar) : Boolean';
27114: Function CharIsOtherSurrogateW( const c : WideChar) : Boolean';
27115: Function CharIsOtherPrivateUseW( const c : WideChar) : Boolean';
27116: Function BitClear( const Bits, BitNo : Integer) : Integer';
27117: Function BitSet( const Bits, BitIndex : Integer) : Integer';
27118: Function BitSetTo( const Bits, BitIndex : Integer; const Value : Boolean) : Integer';
27119: Function BitTest( const Bits, BitIndex : Integer) : Boolean';
27120: Function CharCanonicalCombiningClassW( const Char : WideChar) : Cardinal';
27121: Function CharIsAlphaW( const c : WideChar) : Boolean';
27122: Function CharIsAlphaNumW( const c : WideChar) : Boolean';
27123: Function CharIsCrLf( const c : Char) : Boolean';
27124: Function CharIsCrLfA( const c : AnsiChar) : Boolean';
27125: //Function CharIsCrLfW( const c : WideChar) : Boolean';
27126: Function diCharIsDigit( const c : Char) : Boolean';
27127: Function CharIsDigitA( const c : AnsiChar) : Boolean';
27128: //Function CharIsDigitW( const c : WideChar) : Boolean';
27129: Function CharIsHangulW( const Char : WideChar) : Boolean';
27130: Function CharIsHexDigitW( const c : WideChar) : Boolean';
27131: Function CharIsWhiteSpaceW( const c : WideChar) : Boolean';
27132: Function CharToCaseFoldW( const Char : WideChar) : WideChar';
27133: Function CharToLowerW( const Char : WideChar) : WideChar';
27134: Function CharToUpperW( const Char : WideChar) : WideChar';
27135: Function CharToTitleW( const Char : WideChar) : WideChar';
27136: Function DayOfJulianDate( const JulianDate : TJ JulianDate) : Word';
27137: Function diDayOfWeek( const JulianDate : TJ JulianDate) : Word';
27138: Function DayOfWeekYmd( const Year : Integer; const Month, Day : Word) : Word';
27139: Function diDaysInMonth( const JulianDate : TJ JulianDate) : Word';
27140: Function DaysInMonthYm( const Year : Integer; const Month : Word) : Word';
27141: Procedure DecDay( var Year : Integer; var Month, Day : Word)';
27142: Procedure DecDays( var Year : Integer; var Month, Day : Word; const Days : Integer)';
27143: Function diDeleteDirectory( const Dir : string; const DeleteItself : Boolean) : Boolean';
27144: Function DeleteDirectoryA( Dir : AnsiString; const DeleteItself : Boolean) : Boolean';
27145: //Function DeleteDirectoryW( Dir : UnicodeString; const DeleteItself : Boolean) : Boolean';
27146: Function diEasterSunday( const Year : Integer) : TJ JulianDate';
27147: Procedure EasterSundayYmd( const Year : Integer; out Month, Day : Word)';
27148: Function diFirstDayOfWeek( const JulianDate : TJ JulianDate) : TJ JulianDate';
27149: Procedure FirstDayOfWeekYmd( var Year : Integer; var Month, Day : Word)';
27150: Function diFirstDayOfMonth( const Julian : TJ JulianDate) : TJ JulianDate';
27151: Procedure FirstDayOfMonthYmd( const Year : Integer; const Month : Word; out Day : Word)';
27152: Function diForceDirectories( const Dir : string) : Boolean';
27153: Function ForceDirectoriesA( Dir : AnsiString) : Boolean';
27154: //Function ForceDirectoriesW( Dir : UnicodeString) : Boolean';
27155: Procedure FreeMemAndNil( var Ptr: TObject)'';
27156: Function diGetCurrentFolder : string';
27157: Function GetCurrentFolderA : AnsiString';
27158: //Function GetCurrentFolderW : UnicodeString';
27159: Procedure SetCurrentFolder( const NewFolder : string)';
27160: Procedure SetCurrentFolderA( const NewFolder : AnsiString)';
27161: //Procedure SetCurrentFolderW( const NewFolder : UnicodeString)';
27162: Function diGetDesktopFolder : string';
27163: Function GetDesktopFolderA : AnsiString';
27164: //Function GetDesktopFolderW : UnicodeString';
27165: Function diGetFileSize( const AFileName : string) : Int64';
27166: Function GetFileSizeA( const AFileName : AnsiString) : Int64';
27167: //Function GetFileSizeW( const AFileName : UnicodeString) : Int64';
27168: Function diGetDesktopDirectoryFolder : string';
27169: Function GetDesktopDirectoryFolderA : AnsiString';
27170: //Function GetDesktopDirectoryFolderW : UnicodeString';
27171: Function GetFileLastWriteTime(const FileName: string; out FileTime : TFileTime) : Boolean';
27172: Function GetFileLastWriteTimeA(const FileName: AnsiString; out FileTime: TFileTime): Boolean';
27173: //Function GetFileLastWriteTimeW(const FileName: UnicodeString; out FileTime: TFileTime): Bool';
27174: Function diGetPersonalFolder( const PersonalFolder : Integer) : string';
27175: Function GetPersonalFolderA : AnsiString';
27176: //Function GetPersonalFolderW : UnicodeString';
27177: Function GetSpecialFolder( const SpecialFolder : Integer) : string';
27178: Function GetSpecialFolderA( const SpecialFolder : Integer) : AnsiString';
27179: //Function GetSpecialFolderW( const SpecialFolder : Integer) : UnicodeString';
27180: Procedure diIncMonth( var Year : Integer; var Month, Day : Word)';
27181: Procedure diIncMonths(var Year : Integer; var Month, Day:Word; const NumberOfMonths:Integer);

```

```

27182: Procedure diIncDay( var Year : Integer; var Month, Day : Word');
27183: Procedure IncDays( var Year : Integer; var Month, Day : Word; const Days : Integer');
27184: Function IsDateValid( const Year : Integer; const Month, Day : Word) : Boolean';
27185: Function IsHolidayInGermany( const Julian : TJ JulianDate) : Boolean');
27186: Function IsHolidayInGermanyYmd( const Year : Integer; const Month, Day : Word) : Boolean');
27187: Function diIsLeapYear( const Year : Integer) : Boolean');
27188: Function ISODateToJulianDate( const ISODate : TIsoDate) : TJ JulianDate');
27189: Procedure ISODateToYmd(const ISODate: TIsoDate; out Year : Integer; out Month, Day : Word');
27190: Function IsCharLowLineW( const c : WideChar) : Boolean');
27191: Function IsCharQuoteW( const c : WideChar) : Boolean');
27192: Function IsShiftKeyDown : Boolean');
27193: Function IsCharWhiteSpaceOrAmpersandW( const c : WideChar) : Boolean');
27194: Function IsCharWhiteSpaceOrNoBreakSpaceW( const c : WideChar) : Boolean');
27195: Function IsCharWhiteSpaceOrColonW( const c : WideChar) : Boolean');
27196: Function CharIsWhiteSpaceGtW( const c : WideChar) : Boolean');
27197: Function CharIsWhiteSpaceLtW( const c : WideChar) : Boolean');
27198: Function CharIsWhiteSpaceHyphenW( const c : WideChar) : Boolean');
27199: Function CharIsWhiteSpaceHyphenGtW( const c : WideChar) : Boolean');
27200: Function IsCharWordSeparatorW( const c : WideChar) : Boolean');
27201: Function diISOWeekNumber( const JulianDate : TJ JulianDate) : Word');
27202: Function ISOWeekNumberYmd( const Year : Integer; const Month, Day : Word) : Word');
27203: Function ISOWeekToJulianDate(const Year:Int;const WeekOfYear:Word;const DayOfWeek:Word):TJ JulianDate);
27204: Function JulianDateIsWeekDay( const JulianDate : TJ JulianDate) : Boolean');
27205: Function JulianDateToIsoDate( const Julian : TJ JulianDate) : TIsoDate');
27206: Procedure JulianDateToYmd(const JulianDate:TJ JulianDate;out Year: Integer;out Month,Day:Word);
27207: Function LastDayOfMonth( const JulianDate : TJ JulianDate) : TJ JulianDate');
27208: Procedure LastDayOfMonthYmd( const Year : Integer; const Month : Word; out Day : Word');
27209: Function LastDayOfWeek( const JulianDate : TJ JulianDate) : TJ JulianDate');
27210: Procedure LastDayOfWeekYmd( var Year : Integer; var Month, Day : Word');
27211: Function LastSysErrorMessage : string');
27212: Function LastSysErrorMessageA : AnsiString');
27213: Function LastSysErrorMessageW : UnicodeString');
27214: Function diMax( const a : Integer; const b : Integer) : Integer');
27215: Function diMax3( const a, b, c : Integer) : Integer');
27216: Function MaxCard( const a : Cardinal; const b : Cardinal) : Cardinal');
27217: Function MaxCard3( const a : Cardinal; const b : Cardinal; const c : Cardinal) : Cardinal');
27218: Function dimaxint64( const a : Int64; const b : Int64) : Int64');
27219: Function dimaxint643( const a : Int64; const b : Int64; const c : Int64) : Int64');
27220: Function diMin( const a, b : Integer) : Integer');
27221: Function diMin3( const a, b, c : Integer) : Integer');
27222: Function MinCard( const a, b : Cardinal) : Cardinal');
27223: Function Mincard3( const a, b, c : Cardinal) : Cardinal');
27224: Function diMinInt64( const a, b : Int64) : Int64');
27225: Function diMinint643( const a, b, c : Int64) : Int64');
27226: Function diMinint64U( const a, b : UInt64) : UInt64');
27227: Function diMinint643U( const a, b, c : UInt64) : UInt64');
27228: Function MonthOfJulianDate( const JulianDate : TJ JulianDate) : Word');
27229: Function YearOfJulianDate( const JulianDate : TJ JulianDate) : Integer');
27230: Function YmdToIsoDate( const Year : Integer; const Month, Day : Word) : TIsoDate');
27231: Function YmdToJulianDate( const Year : Integer; const Month, Day : Word) : TJ JulianDate');
27232: end;
27233:
27234: function FibоМaxE2(n: integer): Extended;
27235: begin
27236:   result:= (pow((1+SQRT5)/2,n)-pow((1-SQRT5)/2,n))/SQRT5
27237: end;
27238:
27239:
27240: TDLLVersionInfo=Record
27241:   cbSize, // Size of the structure, in bytes.
27242:   dwMajorVersion, // Major version of the DLL
27243:   dwMinorVersion, // Minor version of the DLL
27244:   dwBuildNumber, // Build number of the DLL
27245:   dwPlatformID: DWord; // Identifies the platform for which the DLL was built
27246: end;
27247:
27248: {$IFDEF MSWINDOWS}
27249: procedure TIdAntiFreeze_Process;
27250: var
27251:   Msg: TMsg;
27252:   ApplicationHasPriority: boolean;
27253: begin
27254:   if ApplicationHasPriority then begin
27255:     Application.ProcessMessages;
27256:   end else begin
27257:     // This guarantees it will not ever call Application.Idle
27258:     if PeekMessage(Msg, 0, 0, PM_NOREMOVE) then begin
27259:       Application.HandleMessage;
27260:     end;
27261:   end;
27262: end;
27263: {$ENDIF}
27264:
27265:
27266: command1:= 'play "'+songpath+'maxbox.wav"'; command2:= 'play "'+songpath+'moon.wav"';
27267: SendMCICommand('open waveaudio shareable'); //parallels
27268: SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav"');
27269: SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav"');
27270: SendMCICommand('close waveaudio');

```

```
27271:  
27272:  
27273: //////////////////////////////////////////////////////////////////  
27274: All maXbox Tutorials Table of Content 2014/2015  
27275: //////////////////////////////////////////////////////////////////  
27276: Tutorial 00 Function-Coding (Blix the Programmer)  
27277: Tutorial 01 Procedural-Coding  
27278: Tutorial 02 OO-Programming  
27279: Tutorial 03 Modular Coding  
27280: Tutorial 04 UML Use Case Coding  
27281: Tutorial 05 Internet Coding  
27282: Tutorial 06 Network Coding  
27283: Tutorial 07 Game Graphics Coding  
27284: Tutorial 08 Operating System Coding  
27285: Tutorial 09 Database Coding  
27286: Tutorial 10 Statistic Coding  
27287: Tutorial 10 Probability Coding  
27288: Tutorial 11 Forms Coding  
27289: Tutorial 12 SQL DB Coding  
27290: Tutorial 13 Crypto Coding  
27291: Tutorial 14 Parallel Coding  
27292: Tutorial 15 Serial RS232 Coding  
27293: Tutorial 16 Event Driven Coding  
27294: Tutorial 17 Web Server Coding  
27295: Tutorial 18 Arduino System Coding  
27296: Tutorial 18_3 RGB LED System Coding  
27297: Tutorial 19 WinCOM /Arduino Coding  
27298: Tutorial 20 Regular Expressions RegEx  
27299: Tutorial 21 Android Coding (coming 2013)  
27300: Tutorial 22 Services Programming  
27301: Tutorial 23 Real Time Systems  
27302: Tutorial 24 Clean Code  
27303: Tutorial 25 maXbox Configuration I+II  
27304: Tutorial 26 Socket Programming with TCP  
27305: Tutorial 27 XML & TreeView  
27306: Tutorial 28 DLL Coding (available)  
27307: Tutorial 29 UML Scripting (2014)  
27308: Tutorial 30 Web of Things (2014)  
27309: Tutorial 31 Closures (2014)  
27310: Tutorial 32 SQL Firebird (2014)  
27311: Tutorial 33 Oscilloscope (available)  
27312: Tutorial 34 GPS Navigation (2014)  
27313: Tutorial 35 Web Box (available)  
27314: Tutorial 36 Unit Testing (avail)  
27315: Tutorial 37 API Coding (avail)  
27316: Tutorial 38 3D Coding (coming 2015)  
27317: Tutorial 39 GEO Map Coding (available)  
27318: Tutorial 39_1 GEO Map Layers Coding (available)  
27319: Tutorial 40 REST Coding (coming 2015)  
27320: Tutorial 41 Big Numbers Coding (2015)  
27321: Tutorial 42 Parallel Processing (2015)  
27322: Tutorial 43 Code Metrics: June2016  
27323: Tutorial 44 IDE Extensions  
27324: Tutorial 45 Robotics: July2016  
27325:  
27326: Doc ref Docu for all Type Class and Const in maXbox_types.pdf  
27327: using Docu for this file is maxbox_functions_all.pdf  
27328: PEP - Pascal Education Program Low Lib Lab ShellHell in UDEMY  
27329:  
27330: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials  
27331: http://stackoverflow.com/tags/pascalscript/hot  
27332: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions  
27333: http://sourceforge.net/projects/maxbox #locs:51620  
27334: http://sourceforge.net/apps/mediawiki/maxbox  
27335: http://www.blaisepascal.eu/  
27336: https://github.com/maxkleiner/maxBox3.git  
27337: http://www.heise.de/download/maxbox-1176464.html  
27338: http://www.softpedia.com/get/Programming/Other-Programming-Files/maxBox.shtml  
27339: https://www.facebook.com/pages/Programming-maXbox/166844836691703  
27340: http://www.softwareschule.ch/arduino_training.pdf  
27341: http://www.delphiarea.com  
27342: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html  
27343: http://entwickler-konferenz.de/2014/speakers/max-kleiner  
27344: http://www.heise.de/download/maxbox-1176464.html  
27345: https://www.udemy.com/learn-coding-from-the-scratch  
27346: http://www.slideshare.net/maxkleiner1/codesign-2015  
27347: https://www.dropbox.com/s/yolconwmq4oqta4/Blaise_2and3_SP_Total.pdf?dl=0  
27348: http://www.softwareschule.ch/download/maxbox_promo.png  
27349: http://max.kleiner.com/maxbox_functions_all.htm  
27350: http://max.kleiner.com/boxart.htm  
27351: http://www.jurgott.org/linkage/util.htm  
27352: http://www.swissdelphicenter.ch/en/niklauswirth.php  
27353: http://www.softwareschule.ch/images/maxbox_20years_delphi.jpg  
27354:  
27355:  
27356: All maXbox Examples List  
27357: https://github.com/maxkleiner/maxbox3/releases  
27358: http://www.softwareschule.ch/download/exampleedition2016.zip  
27359:
```

```

27360: ****
27361: 000_pas_baseconvert.txt
27362: 000_pas_baseconvert.txt_encrypt
27363: 000_pas_baseconvert.txt_decrypt
27364: 001_1_pas_functest - Kopie.txt
27365: 001_1_pas_functest.txt
27366: 001_1_pas_functest2.txt
27367: 001_1_pas_functest_clx2.txt
27368: 001_1_pas_functest_clx2_2.txt
27369: 001_1_pas_functest_openarray.txt
27370: 001_pas_lottogen.txt
27371: 001_pas_lottogen_template.txt
27372: 001_pas_lottogen_txtcopy
27373: 002_pas_russianroulette.txt
27374: 002_pas_russianroulette.txtcopy
27375: 002_pas_russianroulette.txtcopy_decrypt
27376: 002_pas_russianroulette.txtcopy_encrypt
27377: 003_pas_motion.txt
27378: 003_pas_motion.txtcopy
27379: 004_pas_search.txt
27380: 004_pas_search_replace.txt
27381: 004_search_replace_allfunctionlist.txt
27382: 005_pas_oodesign.txt
27383: 005_pas_shellink.txt
27384: 006_pas_oobatch.txt
27385: 007_pas_streamcopy.txt
27386: 008_EINMALEINS_FUNC.TXT
27387: 008_explanation.txt
27388: 008_pas_verwechselt.txt
27389: 008_pas_verwechselt_ibz_bern_func.txt
27390: 008_stack_ibz.TXT
27391: 009_pas_umrunner.txt
27392: 009_pas_umrunner_all.txt
27393: 009_pas_umrunner_componenttest.txt
27394: 009_pas_umrunner_solution.txt
27395: 009_pas_umrunner_solution_2step.txt
27396: 010_pas_oodesign_solution.txt
27397: 011_pas_puzzlepas_defect.txt
27398: 012_pas_umrunner_solution.txt
27399: 012_pas_umrunner_solution2.txt
27400: 013_pas_linenumber.txt
27401: 014_pas_primetest.txt
27402: 014_pas_primetest_first.txt
27403: 014_pas_primetest_sync.txt
27404: 015_pas_designbycontract.txt
27405: 015_pas_designbycontract_solution.txt
27406: 016_pas_searchrec.txt
27407: 017_chartgen.txt
27408: 018_data_simulator.txt
27409: 019_dez_to_bin.txt
27410: 019_dez_to_bin_grenzwert_ibz.txt
27411: 020_proc_feedback.txt
27412: 021_pas_symkey.txt
27413: 021_pas_symkey_solution.txt
27414: 022_pas_filestreams.txt
27415: 023_pas_find_searchrec.txt
27416: 023_pas_pathfind.txt
27417: 024_pas_TFileStream_records.txt
27418: 025_prime_direct.txt
27419: 026_pas_memorystream.txt
27420: 027_pas_shelleexecute_beta.txt
27421: 027_pas_shelleexecute_solution.txt
27422: 028_pas_dataset.txt
27423: 029_pas_assignfile.txt
27424: 029_pas_assignfile_dragndropexe.txt
27425: 030_palindrome_2.txt
27426: 030_palindrome_tester.txt
27427: 030_pas_recursion.txt
27428: 030_pas_recursion2.txt
27429: 031_pas_hashcode.txt
27430: 032_pas_crc_const.txt
27431: 033_pas_cipher.txt
27432: 033_pas_cipher_def.txt
27433: 033_pas_cipher_file_2_solution.txt
27434: 034_pas_soundbox.txt
27435: 035_pas_crcscript.txt
27436: 035_pas_CRCscript_modbus.txt
27437: 036_pas_includetest.txt
27438: 036_pas_includetest_basta.txt
27439: 037_pas_define_demo32.txt
27440: 038_pas_box_demonstrator.txt
27441: 039_pas_dllcall.txt
27442: 040_paspointer.txt
27443: 040_paspointer_old.txt
27444: 041_pasplotter.txt
27445: 041_pasplotter_plus.txt
27446: 042_pas_kgv_ggt.txt
27447: 043_pas_proceduretype.txt
27448: 044_pas_14queens_solwith14.txt
282_fadengraphik.txt
283_SQL_API_messagetimeout.txt
284_SysTools4.txt
285_MineForm_GR32.TXT
285_MineForm_GR32main.TXT
285_MineForm_GR32 mainsolution.TXT
285_MineForm_propas.TXT
285_MineForm_propas2.TXT
285_minesweeper2.TXT
285_Patterns_process.txt
286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
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

```

27449: 044_pas_8queens.txt	309_regex_powertester3.txt
27450: 044_pas_8queens_sol2.txt	310_regex_decorator.TXT
27451: 044_pas_8queens_solutions.txt	312_ListView.txt
27452: 044_queens_performer.txt	313_dmath_dll.txt
27453: 044_queens_performer2.txt	314_fundamentals4_tester.TXT
27454: 044_queens_performer2tester.txt	315_funcplot_dmath.TXT
27455: 045_pas_listhandling.txt	316_cfileutils_cdatetime_tester.TXT
27456: 046_pas_records.txt	317_excel_export_tester.TXT
27457: 047_pas_modula10.txt	318_excel_export.TXT
27458: 048_pas_romans.txt	318_excel_export2.TXT
27459: 049_pas_ifdemo.txt	318_excel_export3.TXT
27460: 049_pas_ifdemo_BROKER.txt	318_excel_export3_tester.TXT
27461: 050_pas_primetest2.txt	319_supergeneral_math.TXT
27462: 050_pas_primetester_thieves.txt	319_supergeneral_mathdefect.TXT
27463: 050_program_starter.txt	320_supergeneral.TXT
27464: 050_program_starter_performance.txt	320_supergeneral2.TXT
27465: 051_pas_findtext_solution.txt	321_SQL_Excel.txt
27466: 052_pas_text_as_stream.txt	321_SQL_Excel2.txt
27467: 052_pas_text_as_stream_include.txt	321_SQL_Excel_Export.txt
27468: 053_pas_singleton.txt	321_SQL_ExportExec.txt
27469: 054_pas_speakpassword.txt	321_SQL_ExportTest.txt
27470: 054_pas_speakpassword2.txt	321_SQL_SAS_tester3.txt
27471: 054_pas_speakpassword_searchtest.txt	321_SQL_SAS_tester3_selfcompile.txt
27472: 055_pas_factorylist.txt	321_SQL_SAS_tester3_selfcompile2.txt
27473: 056_pas_demeter.txt	321_SQL_SAS_tester4.txt
27474: 057_pas_dirfinder.txt	321_SQL_SAS_updater.txt
27475: 058_pas_filefinder.txt	322_timezones.TXT
27476: 058_pas_filefinder_pdf.txt	323_datefind_fulltext_search.txt
27477: 058_pas_filefinder_screview.txt	323_datefind_fulltext_searchtester.txt
27478: 058_pas_filefinder_screview2.txt	324_interfacenavi.TXT
27479: 058_pas_filefinder_screview3.txt	325_ampelsteuerung.txt
27480: 059_pas_timertest.txt	325_analogclock.txt
27481: 059_pas_timertest_2.txt	326_world_analogclock.txt
27482: 059_pas_timertest_time_solution.txt	326_world_analogclock2.txt
27483: 059_timerobject_starter2.txt	327_atomimage_clock.txt
27484: 059_timerobject_starter2_ibz2_async.txt	328_starfield.txt
27485: 059_timerobject_starter2_uml.txt	329_starfield2.txt
27486: 059_timerobject_starter2_uml_main.txt	330_myclock.txt
27487: 059_timerobject_starter4_ibz.txt	330_myclock2.txt
27488: 060_pas_datefind.txt	331_SQL_DBfirebird4.txt
27489: 060_pas_datefind_exceptions2.txt	332_jprofiler.txt
27490: 060_pas_datefind_exceptions_CHECKTEST.txt	332_jprofiler_form.txt
27491: 060_pas_datefind_fulltext.txt	332_jprofiler_form2.txt
27492: 060_pas_datefind_plus.txt	333_querybyexample.txt
27493: 060_pas_datefind_plus_mydate.txt	333_querybyexample2.txt
27494: 061_pas_randomwalk.txt	334_jvutils_u.txt
27495: 061_pas_randomwalk_plus.txt	335_atomimage5.txt
27496: 062_pas_korrelation.txt	335_atomimage6.txt
27497: 063_pas_calculateform.txt	335_atomimage7.txt
27498: 063_pas_calculateform_2list.txt	336_digiclock.txt
27499: 064_pas_timertest.txt	336_digiclock2.txt
27500: 065_pas_bitcounter.txt	336_digiclock2test.txt
27501: 066_pas_eliza.txt	336_digiclock3.txt
27502: 066_pas_eliza_include_sol.txt	337_4games.txt
27503: 067_pas_morse.txt	337_4games_inone.txt
27504: 068_pas_piezo_sound.txt	338_compress.txt
27505: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT	338_compress2.txt
27506: 069_my_LEDBOX.TXT	339_ntfs.txt
27507: 069_pas_ledmatrix.txt	340_docutype.txt
27508: 069_pas_LEDMATRIX_Alphabet.txt	340_logsimulation.txt
27509: 069_pas_LEDMATRIX_Alphabet_run.txt	340_logsimulation2.txt
27510: 069_pas_LEDMATRIX_Alphabet_tester.txt	340_soundControltype.txt
27511: 069_PAS_LEDATRIX_COLOR.TXT	341_blix_clock.txt
27512: 069_pas_ledmatrix_fixedit.txt	341_blix_clock2.txt
27513: 069_pas_LEDATRIX_soundbox.txt	341_blix_clock_tester.txt
27514: 069_pas_LEDATRIX_soundbox2.txt	342_set_enumerator.txt
27515: 069_Richter_MATRIX.TXT	343_dice2.txt
27516: 070_pas_functionplot.txt	344_pe_header.txt
27517: 070_pas_functionplotter2.txt	344_pe_header2.txt
27518: 070_pas_functionplotter2_mx4.txt	345_velocity.txt
27519: 070_pas_functionplotter2_tester.txt	346_conversions.txt
27520: 070_pas_functionplotter3.txt	347_pictureview.txt
27521: 070_pas_functionplotter4.txt	348_duallistview.txt
27522: 070_pas_functionplotter_digital.txt	349_biginteger.txt
27523: 070_pas_functionplotter_elliptic.txt	350_parserform.txt
27524: 070_pas_function_helmholtz.txt	351_chartform.txt
27525: 070_pas_textcheck_experimental.txt	351_chartform2.txt
27526: 071_pas_graphics.txt	351_chartform3.txt
27527: 071_pas_graphics_drawsym.txt	352_array_unittest.txt
27528: 071_pas_graphics_drawsym_save.txt	353_smtp_email.txt
27529: 071_pas_graphics_random.txt	353_smtp_email2.txt
27530: 072_pas_fractals.txt	354_josephus.txt
27531: 072_pas_fractals_2.txt	355_life_of_PI.txt
27532: 072_pas_fractals_blackhole.txt	356_3D_printer.txt
27533: 072_pas_fractals_perfomance.txt	357_fplot.TXT
27534: 072_pas_fractals_perfomance_new.txt	358_makesound.txt
27535: 072_pas_fractals_perfomance_sharp.txt	359_charsetrules.TXT
27536: 072_pas_fractals_performance.txt	360_allobjects.TXT
27537: 072_pas_fractals_performance_mx4.txt	360_JvPaintFX.TXT

```

27538: 073_pas_forms.txt
27539: 074_pas_chartgenerator.txt
27540: 074_pas_chartgenerator_solution.txt
27541: 074_pas_chartgenerator_solution_back.txt
27542: 074_pas_charts.txt
27543: 075_bitmap_Artwork2.txt
27544: 075_pas_bitmappuzzle.txt
27545: 075_pas_bitmappuzzle24.prod.txt
27546: 075_pas_bitmappuzzle2_prod.txt
27547: 075_pas_bitmappuzzle3.txt
27548: 075_pas_bitmapsolve.txt
27549: 075_pas_bitmap_Artwork.txt
27550: 075_pas_puzzlespas_solution.txt
27551: 076_pas_3dcube.txt
27552: 076_pas_circle.txt
27553: 077_pas_mmshow.txt
27554: 078_pas_pi.txt
27555: 079_pas_3dcube_animation.txt
27556: 079_pas_3dcube_animation4.txt
27557: 079_pas_3dcube_plus.txt
27558: 080_pas_hanoi.txt
27559: 080_pas_hanoi2.txt
27560: 080_pas_hanoi2_file.txt
27561: 080_pas_hanoi2_sol.txt
27562: 080_pas_hanoi2_tester.txt
27563: 080_pas_hanoi2_tester_fast.txt
27564: 080_pas_hanoi3.txt
27565: 081_pas_chartist2.txt
27566: 082_pas_biorhythmus.txt
27567: 082_pas_biorhythmus_solution.txt
27568: 082_pas_biorhythmus_solution_3.txt
27569: 082_pas_biorhythmus_test.txt
27570: 083_pas_GITARRE.txt
27571: 083_pas_soundbox_tones.txt
27572: 084_pas_waves.txt
27573: 085_mxsinus_logo.txt
27574: 085_sinus_plot_waves.txt
27575: 086_pas_graph_arrow_heart.txt
27576: 087_bitmap_loader.txt
27577: 087_pas_bitmap_solution.txt
27578: 087_pas_bitmap_solution2.txt
27579: 087_pas_bitmap_subimage.txt
27580: 087_pas_bitmap_test.txt
27581: 088_pas_soundbox2_mp3.txt
27582: 088_pas_soundbox_mp3.txt
27583: 088_pas_sphere_2.txt
27584: 089_pas_gradient.txt
27585: 089_pas_maxland2.txt
27586: 090_pas_sudoku4.txt
27587: 090_pas_sudoku4_2.txt
27588: 091_pas_cube4.txt
27589: 092_pas_statistics4.txt
27590: 093_variance.txt
27591: 093_variance_debug.txt
27592: 094_pas_daysold.txt
27593: 094_pas_stat_date.txt
27594: 095_pas_ki_simulation.txt
27595: 096_pas_geisen_problem.txt
27596: 096_pas_montyhall_problem.txt
27597: 097_lotto_proofofconcept.txt
27598: 097_pas_lottocombinations_beat_plus.txt
27599: 097_pas_lottocombinations_beat_plus2.txt
27600: 097_pas_lottocombinations_universal.txt
27601: 097_pas_lottosimulation.txt
27602: 098_pas_chartgenerator_plus.txt
27603: 099_pas_3D_show.txt
27604: 200_big_numbers.txt
27605: 200_big_numbers2.txt
27606: 201_streamload_xml.txt
27607: 202_systemcheck.txt
27608: 203_webservice_simple_intftester.txt
27609: 204_webservice_simple.txt
27610: 205_future_value_service.txt
27611: 206_DTD_string_functions.txt
27612: 207_ibz2_async_process.txt
27613: 208_crc32_hash.txt
27614: 209_cryptohash.txt
27615: 210_public_private.txt
27616: 210_public_private_cryptosystem.txt
27617: 211_wipe_pattern.txt
27618: 211_wipe_pattern2.txt
27619: 211_wipe_pattern_solution.txt
27620: 212_pas_statisticmodule4.TXT
27621: 212_pas_statisticmodule4.TXT
27622: 212_statisticmodule4.txt
27623: 213_pas_BBP_Algo.txt
27624: 214_mxdocudemo.txt
27625: 214_mxdocudemo2.txt
27626: 214_mxdocudemo3.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_syndit.txt
370_syndit2.txt
370_syndit2_mxtester.txt
370_syndit2_mxtester2.txt
371_maxbook_v4tester.txt
372_stackibz2_memoryalloc.TXT
372_syndit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefuncTesterfilemon.txt
380_coolfunc.txt
380_coolfunc2.txt
380_coolfunc_tester.txt
381_bitcoin_simulation.txt
382_GRMath.TXT
382_GRMath_PI_Proof.TXT
382_GRMath_Riemann.TXT
383_MDAC_DCOM.txt
384_TeamViewerID.TXT
386_InternetRadio.TXT
387_fulltextfinder.txt
387_fulltextfinder_cleancode.txt
387_fulltextfinder_fast.txt
387_fulltext_getscripttest.txt
388_TCPServerSock.TXT
388_TCPServerSock2.TXT
388_TCPServerSockClient.TXT
389_TAR_Archive.TXT
389_TAR_Archive_test.TXT
389_TAR_Archive_test2.TXT
390_Callback3.TXT
390_Callback3Rec.TXT
390_CallbackClean.TXT
390_StringlistHTML.TXT
391_ToDo_List.TXT
392_Barcode.TXT
392_Barcode2.TXT
392_Barcode23.TXT
392_Barcode2scholz.TXT
392_Barcode3scholz.TXT
393_QRCode.TXT
393_QRCode2.TXT
393_QRCode2Direct.TXT
393_QRCode2DirectIndy.TXT
393_QRCode2Direct_detlef.TXT
393_QRCode3.TXT
394_networkgraph.TXT
394_networkgraph_depwalkutilstest.TXT
394_networkgraph_depwalkutilstest2.TXT
395_USBController.TXT
396_Sort.TXT
397_Hotlog.TXT
397_Hotlog2.TXT
398_ustrings.txt
399_form_templates.txt

```

```

27627: 215_hints_test.TXT
27628: 216_warnings_test.TXT
27629: 217_pas_heartbeat.txt
27630: 218_biorhythmus_studio.txt
27631: 219_cipherbox.txt
27632: 219_crypt_source_comtest_solution.TXT
27633: 220_cipherbox_form.txt
27634: 220_cipherbox_form2.txt
27635: 221_bcd_explain.txt
27636: 222_memoform.txt
27637: 223_directorybox.txt
27638: 224_dialogs.txt
27639: 225_sprite_animation.txt
27640: 226_ASCII_Grid2.TXT
27641: 227_animation.txt
27642: 227_animation2.txt
27643: 228_android_calendar.txt
27644: 229_android_game.txt
27645: 229_android_game_tester.txt
27646: 230_DataProvider.txt
27647: 230_DataSetProvider.txt
27648: 230_DataSetXMLBackupScholz.txt
27649: 231_DBGrid_access.txt
27650: 231_DBGrid_XMLaccess.txt
27651: 231_DBGrid_XMLaccess2.txt
27652: 231_DBGrid_XMLaccess_locatetester.txt
27653: 231_DBGrid_XML_CDS_local.txt
27654: 232_outline.txt
27655: 232_outline_2.txt
27656: 233_modular_form.txt
27657: 234_debugoutform.txt
27658: 235_fastform.TXT
27659: 236_componentpower.txt
27660: 236_componentpower_back.txt
27661: 237_pas_4forms.txt
27662: 238_lottogen_form.txt
27663: 239_pas_sierpinski.txt
27664: 239_pas_sierpinski2.txt
27665: 240_unitGlobal_tester.txt
27666: 241_db3_sql_tutorial.txt
27667: 241_db3_sql_tutorial2.txt
27668: 241_db3_sql_tutorial2fix.txt
27669: 241_db3_sql_tutorial3.txt
27670: 241_db3_sql_tutorial3connect.txt
27671: 241_db3_sql_tutorial3_ftest.txt
27672: 241_RTL_SET2.txt
27673: 241_RTL_SET2_tester.txt
27674: 242_Component_Control.txt
27675: 243_tutorial_loader.txt
27676: 244_script_loader_loop.txt
27677: 245_formapp2.txt
27678: 245_formapp2_tester.txt
27679: 245_formapp2_testerX.txt
27680: 246_httpapp.txt
27681: 247_datecalendar.txt
27682: 248_ASCII_Grid2_sorted.TXT
27683: 249_picture_grid.TXT
27684: 250_tipsandtricks2.txt
27685: 250_tipsandtricks3.txt
27686: 250_tipsandtricks3api.txt
27687: 250_tipsandtricks3_admin_elevation.txt
27688: 250_tipsandtricks3_tester.txt
27689: 250_tipsandtricks4_tester.txt
27690: 250_tipsandtricks4_tester2.txt
27691: 251_compare_noise_gauss.txt
27692: 251_whitenoise.txt
27693: 251_whitenoise2.txt
27694: 252_hilbert_turtle.txt
27695: 252_pas_hilbert.txt
27696: 253_opearatingsystem3.txt
27697: 254_dynarrays.txt
27698: 255_einstein.txt
27699: 256_findconsts_of_EXE.txt
27700: 256_findfunctions2_of_EXE.txt
27701: 256_findfunctions2_of_EXEaverp.txt
27702: 256_findfunctions2_of_EXEspec.txt
27703: 256_findfunctions3.txt
27704: 256_findfunctions_of_EXE.txt
27705: 257_AES_Cipher.txt
27706: 258_AES_cryptobox.txt
27707: 258_AES_cryptobox2.txt
27708: 258_AES_cryptobox2_passdlg.txt
27709: 259_AES_crypt_directory.txt
27710: 260_sendmessage_2.TXT
27711: 260_sendmessage_beta.TXT
27712: 261_probability.txt
27713: 262_mxoutputdemo4.txt
27714: 263_async_sound.txt
27715: 264_vclutils.txt

400_fploottchart.TXT
400_fploottchart2.TXT
400_fploottchart2teetest.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_kryptonum.TXT
424_opengl_tester.txt
425_reversi_game.txt
426_IBUtils.TXT
427_IBDatabase.TXT
428_SortGrid.TXT
429_fileclass.txt
430_fileoperation.txt
430_fileoperation_tester.txt
431_performance_index.txt
432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
437_WinAPItop_Firebirdtester.txt
438_OvciInternational.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

```

```

27716: 264_VCL_utils2.txt          457_ressource_grid.txt
27717: 265_timer_API.txt          458_atomimageX.txt
27718: 266_serial_interface.txt   459_cindyfunc.txt
27719: 266_serial_interface2.txt  459_cindyfunc2.txt
27720: 266_serial_interface3.txt  460_TopTenFunctions.txt
27721: 267_ackermann_rec.txt     461_sqlform_calwin.txt
27722: 267_ackermann_variants.txt 462_caesarcipher.txt
27723: 268_DBGrid_tree.txt       463_global_exception.txt
27724: 269_record_grid.TXT       464_function_procedure.txt
27725: 270_Jedi_FunctionPower.txt 464_function_procedure2.txt
27726: 270_Jedi_FunctionPowerTester.txt 464_function_procedure3.txt
27727: 271_closures_study.txt    465_U_HexView.txt
27728: 271_closures_study_workingset2.txt 466_moon.txt
27729: 272_pas_function_show.txt 466_moon_inputquery.txt
27730: 273_pas_function_show2.txt 4671_cardmagic.txt
27731: 274_library_functions.txt 467_helmholtz_graphic.txt
27732: 275_turtle_language.txt    468_URLMon.txt
27733: 275_turtle_language_save.txt 468_URLMon2.txt
27734: 276_save_algo.txt         469_formarrow.txt
27735: 276_save_algo2.txt        469_formarrow_datepicker.txt
27736: 277_functionsfor39.txt   469_formarrow_datepicker_ibz_result.txt
27737: 278_DB_Dialogs.TXT       469_ibzresult.txt
27738: 279_hexer2.TXT          470_DFFUtils_compiled.txt
27739: 279_hexer2macro.TXT      470_DFFUtils_ScrollingLED.txt
27740: 279_hexer2macroback.TXT   470_Oscilloscope.txt
27741: 280_UML_process.txt      470_Oscilloscope_code.txt
27742: 280_UML_process_knabe2.txt 471_cardmagic.txt
27743: 280_UML_process_knabe3.txt 471_cardmagic2.txt
27744: 280_UML_process_TIM_Botzenhardt.txt 472_allcards.TXT
27745: 280_UML_TIM_Seitz.txt    473_comboset.txt
27746: 281_picturepuzzle.txt    474_wakeonlan.txt
27747: 281_picturepuzzle2.txt   474_wakeonlan2.txt
27748: 281_picturepuzzle3.txt   476_getscripttest.txt
27749: 281_picturepuzzle4.txt   477_filenameonly.txt
27750: 479_inputquery.txt        480_regex_pathfinder.txt
27751: 480_regex_pathfinder2.txt 481_processList.txt
27752: 482_processPipe.txt      482_processPipeGCC.txt
27753: 483_PathFuncTest_mx.txt  484_filefinder3.txt
27754: 485_InnoFunc.txt        486_VideoGrabber.txt
27755: 487_asyncKeyState.txt    488_asyncTerminal.txt
27756: 489_simpleComport.txt   490_webCamproc.txt
27757: 491_analogmeter.txt     492_snowflake2.txt
27758: 493_gadgets.txt         495_fourierfreq.txt
27759: 496_InstallX.txt        497_LED.txt
27760: 498_UnitTesting.txt      499_mulu42.txt
27761: 500_diceoflifes.txt      501_firebird_datasnap_tests.txt
27762: 502_findalldocs.txt      503_led_switch.txt
27763: 504_fileclass.txt       505_debug.txt
27764: 506_colormatrix.txt     507_derutils.txt
27765: 508_simplecomportmorse.txt 509_GEOMap2.txt
27766: 509_509_GEOMap2_SReverse.TXT 510_510_bonn_gpsdata_mx4.pas
27767: 511_LEDLabel.txt         512_LED_moon.txt
27768: 513_StreamIntegration.txt 514_LED_moon2.txt
27769: 515_ledclock3.txt       516_mapview.txt
27770: 517_animation7.txt      518_sensors_meter.txt
27771: 519_powtills.txt        520_run bytecode.txt
27772: 521_iputils2.txt        522_getgeocode.txt
27773: 523_NMEA.txt           524_NAV_Utils.txt
27774: 525_GEO84s.txt         526_Compass_meter.txt
27775: 527_GPSDemo.txt        528_linescount.txt
27776: 529_profilertest.txt   530_3DLab.txt
27777: 531_profilertest.txt   532_mcicommand.txt
27778: 533_syncasync_demo.txt  534_arduino_cockpit.TXT
27779: 535_Battleship3.pas     536_ressource_grid2.txt
27780: 537_iniplus.TXT        538_shellbatch.txt
27781: 539_timeturtle123.txt   540_NeuralNetwork.pas
27782: 541_webserver_arduino_motorturtle.txt 542_arduino_sound.txt
27783: 543_MATH_TurboP.PAS     544_UTIL01.PAS
27784: 545_strips.TXT          546_fourier3.pas
27785: 547_regexmaster.TXT     548_STExpressions.TXT -Services
27786: 549_3D_Panorama.txt     550_Expressions.TXT - 550_ADO_OLEDB.txt
27787: 551_ArduinoTester.txt   552_WaitExec32.txt
27788: 553_ArduinoCockBit3.txt  554_Watdchdog.txt - 555_CODEsign2.txt
27789: 556_stringlistrandom.TXT 557_4dice2015.txt
27790: 558_highrestimer.TXT   -559_highthrestimer2
27791: 561_newfunctions399160.txt 560_PSUtils.TXT
27792: 563_moonpaper.txt       562_shellctrldemo.txt
27793: 565_ConsoleCapture.txt   564_queryperformance.txt
27794: 567_SquareWordGrids2.txt 566_queryperformance2.txt
27795: 569_keylog.txt - 569_ServiceMgr2.TXT 568_U_BigFloatTestsScript2.pas
27796: 571_myPing.txt          570_turingspeech.txt
27797: 573_modbusfrm_Main.pas   572_shellctrlplus.txt
27798: 575_TARTARUGA/Desktop.txt 574_arduino_cockpit5.TXT
27799: 577_listbox2list.txt    576_outlineEX11.PAS
27800: 579_numbersystems_sort.txt 578_access_db_logsimulation3.txt
27801: 581_stringstream.txt    580_indystacksearch_geo.txt
27802: 583_VirtualConstructor_savereport.txt 582_indystackwin.txt
27803: 585_fulltextfinder_cleancode_override.txt 584_ProcessList2fontwidth.txt last
27804: 587_one_function.txt    586_STRandom.txt
                                         588_XSBuiltins.txt

```

```

27805: 589_avi_animate.txt
27806: 591_emailattach.txt
27807: 593_round_time.txt
27808: 595_check_memory.txt
27809: 597_ole_commands.txt
27810: 599_bug.txt
27811: 601_PECheckSum.txt
27812: 603_cupids_arrow.TXT
27813: 605_maxonmotor3DTage2BASTA2015.TXT
27814: 607_DataSetProvider_CDS_ADO.txt
27815: 609_ScriptExecutor (beta)
27816: 611_Arduino_COMOutputs.txt
27817: 613_uPSI_DIUtils_test.pas
27818: 615 SONAR_51_Starter.txt
27819:
27820: (589_AVI_Throbber.res.txt)
27821: 616_ComTerminalDlg.pas
27822:
27823: 617_API_coding_tut37.txt
27824: 618_bayes_filter.txt
27825: 618_bayes_filter_dic2.txt
27826: 619_crypto_package_demo_policy.txt
27827: 620_serialtimer.txt
27828: 622_netsh_master_restore4.psb
27829: 623_exspawnu.pas
27830: 624_Miscstest_lexscanner.pas
27831: 625_binomial_bigint3 testcase.txt
27832: 627_4gewinnt_main3.txt
27833: 629_Word_FORMULA.PAS
27834: 630_multikernel3.TXT
27835: 632_safecall112.txt
27836: 632_safecheck2_regex_basta.txt
27837: 634_briefcase_XMLDB.xml
27838: 634_maxbase3server.mdb
27839: 636_rest_apiexamples.txt
27840: 637_psnconvert4.TXT
27841: 638_Mathslib_prime.pas.txt
27842: 640_rest_geocode.txt
27843: 640_weather_cockpit5.TXT
27844: 641_voicecommand.TXT
27845: 643_EKON19_Tester3_OLE_Reg.TXT
27846: 645_age_guess_REX.pas
27847: 646_pi_evil2.TXT
27848: 648_SYNEdit_ExportDemo.TXT
27849: 649_StringlistHTMLtestBasta.TXT
27850: 650_http_server2tester.txt
27851: 650_dbblober.txt
27852: 651_StrUtilmaxbase.pas
27853: 653_Pdoxtstbox.pas
27854: 655_arduino_chess.txt
27855: 656_XML_RpcCommon.pas
27856: 657_PEM_Cert.txt
27857: 659_VirListBox.txt
27858: 660_InetUtils_Rest.txt
27859: 662_VisualChronForm.pas
27860: 664_DataAnnotationsValid.pas.pas
27861: 666_SysInfoCtrls.pas.pas
27862: 667_URobo2.pas.pas
27863: 669_uptime.txt.pas
27864: 670_interesting_birthday_paradox.txt
27865: 672_regex_ask_task.txt
27866: 674_pipe_graphics.txt
27867: 676_run_as_admin.txt
27868: 678_fmTraceRouteMainU.pas
27869: 680_gravity_waves.txt
27870: 682_ScreenMatrixToBitmap.pas
27871: 684_mathe4.pas
27872: 686_namedpipe_demo.txt
27873: 687_memorymapped_filestream.txt
27874: 689_proctype_alias.txt
27875: 691_httpservermain3.txt
27876: 692_imageserv_fClient33.pas
27877: 694_OverbyteIcsAsnlUtils.pas
27878: 696_kmemo_demo_Main.pas
27879: 697_OverbyteIcsTicks64.pas
27880: 698_new_classes_tester.pas
27881: 698_Bitcoin_blockchaintester.pas
27882: 700_mx422_functions_demo2.txt
27883: 700_new_function_snippets.pas
27884: 701_all_fibonacci.txt
27885: 703_pc_sensors.txt
27886: 705_magsubsl_functions.pas
27887: 706_population_count.txt
27888: 708_XML_DOM_ReaderTutor.txt
27889: 710_enigma_riddler.txt
27890: 710_geo_distance_form_mapbox.txt
27891: 710_RegSrvUtils_mx4.pas
27892: 712_rootsfunction.txt
27893: 713_SnakeA2.pas

590_HTML_to_RTF.txt
592_getTypeLibList.txt
594_check_creditcard.txt
596_time_delays.txt
598_software_list.txt
600_surprise_nice.txt
602_multilang_game.txt (moonbug)
604_GEOCodeReverse4.TXT
606_U_FibonacciSunflower.TXT
608_ColorMixer_Arduino2.txt
610_3D_DLL.txt
612_MTterminal.pas
614_inbrowserock.txt
616_ComTerminalDlg.pas

589_avi_animate.txt
//end for 3.9.9.195

617_API_coding_tut39.txt
618_bayes_filter_dic.txt
619_crypto_package_demo.txt
619_crypto_packr.txt
621_bruteforceattack_dic.txt
622_netsh_master_sonar.pas
623_exspawnu_buffoverflow.pas
624_Miscstest_lexscanner2.pas
626_lorentztransformation.TXT
628_FormatFloat.pas
630_multikernel2.TXT
631_dmath_testfunc_mx3.pas
632_safecheck2.txt
633_gamma_func.txt
634_BriefcaseMain2.pas
635_bitcount_bytocodestudy.txt
637_psnconvert3.TXT
637_startscript_psnconvert2.TXT
639_prime_factors.txt
640_rest_weather_report.txt
640_weather_cockpit6.TXT
642_BDE_export.PAS
644_Synapse_Mimetest_regEx3.txt
646_pi_evil.TXT
647_StringtoHTMLBasta.TXT
649_dayio.pas
650_sched.pas
650_time_routines.txt
650_drawdice2016.txt
652_graph3DMainUnit2.pas
654_spectrum_dice2016_3.txt
655_arduino_chess2.txt
656_XML_RpcCommon2.pas
658 ASN1_Cert.txt
660_X509_Cert.pas
661_REXX.pas
663_Tokens.pas
665_keditcommon.pas.pas
667_URobot1.pas.pas
668_U_SoundGen3.pas.pas
670_interestingX.txt
671_replacedigit.txt //File
673_pipe_singapure.txt //LAB
675_bitcoin_doublehash.txt
677_pingtest4.txt
679_messagefMain.pas
681_kronos_3DLab.TXT
683_mx422_functions_demo.txt
685_leanfitmath.txt
686_namedpipe_demo2.txt
688_KGraphics_Call.txt
690_calc_pi_ex.pas
692_fsveremain3.pas
693_OverbyteIcsUtils_tester.pas
695_IdAntiFreeze_tester.pas
696_kmemo_demo.pas
697_OverbyteIcsTicks64Demo1.pas
693_OverbyteIcsDnsQuerytester.pas
699_OverbyteIcsShal_tester.pas
700_mx422_functions_demo3.txt
700_function_snippets3_metrics.inc
702_pascal_fibonacci.txt
704_entropy_test.txt
705_parser_demo_case.txt
707_GrayCode.txt
709_BarcodeReader.txt
710_EnumRegKeys.txt
710_geo_distance_dialog.txt
711_geo_satellite_mapbox.txt
712_towerofhanoi_animation.pas
713_internet_traces_mx_screenshot2.png

```

```

27894: 714_astroids_GAME3.PAS
27895: 715_shape_calculator.PAS
27896: 717_AdLog.PAS
27897: 719_rootsfunction.txt
27898: 720_HTML_TableExport_CSS.pas
27899: 721_DOSCapture_netstat2.TXT
27900: 723_remote_notepad_message.TXT
27901: 725_CRC32_live.txt
27902: 725_725_Cholesky.pas
27903: 725_blockchain_sensors3.txt
27904:
27905:
27906: http://max.kleiner.com/boxart.htm
27907:
27908: Proposal for Boolean Logic Abbreviation Symbol:
27909: -----
27910:
27911: 01 FALSE //Contradiction
27912:
27913: 02 AND //Conjunction  $x \cdot y$ 
27914:
27915: 03 INHIB //Inhibition  $x^{\neg}y$ 
27916:
27917: 04 PRAEP //Praependence  $x$ 
27918:
27919: 05 PRAE //Praesection  $\neg x \cdot y$ 
27920:
27921: 06 POST //Postpendence  $y$ 
27922:
27923: 07 XOR //Exclusive OR  $x^{\neg}y + \neg x \cdot y$ 
27924:
27925: 08 OR //Disjunction OR =  $x + y$ 
27926:
27927: 09 NOR //Rejection
27928:
27929: 10 AEQ //Aequivalence  $x \leftrightarrow y, \neg x \cdot \neg y + x \cdot y$ 
27930:
27931: 11 NEGY //YNegation  $\neg y$ 
27932:
27933: 12 IMPY //YImplication  $y \rightarrow x; \neg y \cdot x$ 
27934:
27935: 13 NEGX //Xnegation  $\neg x$ 
27936:
27937: 14 IMPX //XImplication  $x \rightarrow y; \neg x \cdot y$ 
27938:
27939: 15 NAND //Exclusion
27940:
27941: 16 TRUE //TRUE Tautologic
27942: -----
27943:
27944:
27945: http://sourceforge.net/projects/maxbox/files/Examples/
27946: http://sourceforge.net/projects/maxbox/files/Examples/
27947:
27948: Help Online:
27949: http://max.kleiner.com/maxbox_functions_all.htm
27950:
27951: WebScript Examples:
27952:
27953: http://www.softwareschule.ch/examples/performer.txt;
27954: http://www.softwareschule.ch/examples/turtle.txt;
27955: http://www.softwareschule.ch/examples/SQLExport.txt;
27956: http://www.softwareschule.ch/examples/Richter.txt;
27957: http://www.softwareschule.ch/examples/checker.txt;
27958: http://www.softwareschule.ch/examples/demoscript.txt;
27959: http://www.softwareschule.ch/examples/ibzresult.txt;
27960: http://www.softwareschule.ch/examples/performindex.txt
27961: http://www.softwareschule.ch/examples/processlist.txt
27962: http://www.softwareschule.ch/examples/game.txt
27963: http://www.softwareschule.ch/examples/GEOGPS.txt
27964: http://www.softwareschule.ch/examples/turtle2.txt
27965: http://www.softwareschule.ch/examples/turtle3.txt
27966: http://www.softwareschule.ch/examples/asyncterminal.txt
27967: http://www.softwareschule.ch/examples/snowflake.txt
27968: http://www.softwareschule.ch/examples/arduinoled.txt
27969: http://www.softwareschule.ch/examples/moon2.txt
27970: http://www.softwareschule.ch/examples/cockpit.txt
27971: http://www.softwareschule.ch/examples/tartaruga.txt
27972: http://www.softwareschule.ch/examples/surprise.txt
27973: http://www.softwareschule.ch/examples/weatherapp2.txt
27974: http://www.softwareschule.ch/examples/bigint.txt
27975:
27976:
27977: Delphi Basics Run Time Library listing
27978: ****
27979: A
27980: Compiler Directive $A Determines whether data is aligned or packed
27981: Compiler Directive $Align Determines whether data is aligned or packed
27982: Compiler Directive $AppType Determines the application type : GUI or Console

```

27983: **Procedure** SysUtils Abort Aborts the current processing **with** a silent exception  
 27984: **Function** System Abs Gives the **absolute** value **of** a number (**-ve** sign **is** removed)  
 27985: Directive **Abstract** Defines a **class** method only implemented **in** subclasses  
 27986: Variable System AbstractErrorProc Defines a proc called when an **abstract** method **is** called  
 27987: **Function** System Addr Gives the address **of** a variable, **function or procedure**  
 27988: Keyword **And** Boolean **and** or bitwise **and** of two arguments  
 27989: **Type** System AnsiChar A character **type** guaranteed **to** be 8 bits **in** size  
 27990: **Function** SysUtils AnsiCompareStr Compare two strings **for** equality  
 27991: **Function** SysUtils AnsiCompareText Compare two strings **for** equality, ignoring **case**  
 27992: **Function** StrUtils AnsiContainsStr Returns true **if** a **string** contains a substring  
 27993: **Function** StrUtils AnsiEndsStr Returns true **if** a **string** ends **with** a substring  
 27994: **Function** StrUtils AnsiIndexStr Compares a **string** with a list **of** strings - returns match index  
 27995: **Function** StrUtils AnsiLeftStr Extracts characters from the left **of** a **string**  
 27996: **Function** SysUtils AnsiLowerCase Change upper **case** characters **in** a **string** to lower **case**  
 27997: **Function** StrUtils AnsiMatchStr Returns true **if** a **string** exactly matches one **of** a list **of** strings  
 27998: **Function** StrUtils AnsiMidStr Returns a substring from the middle characters **of** a **string**  
 27999: **Function** StrUtils AnsiPos Find the position **of** one **string** **in** another  
 28000: **Function** StrUtils AnsiReplaceStr Replaces a part **of** one **string** **with** another  
 28001: **Function** StrUtils AnsiReverseString Reverses the sequence **of** letters **in** a **string**  
 28002: **Function** StrUtils AnsiRightStr Extracts characters from the right **of** a **string**  
 28003: **Function** StrUtils AnsiStartsStr Returns true **if** a **string** starts **with** a substring  
 28004: **Type** System AnsiString A data **type** that holds a **string** **of** AnsiChars  
 28005: **Function** SysUtils AnsiUpperCase Change lower **case** characters **in** a **string** to upper **case**  
 28006: **Procedure** System Append Open a text **file** **to** allow appending **of** text **to** the **end**  
 28007: **Procedure** SysUtils AppendStr Concatenate one **string** onto the **end** **of** another  
 28008: **Function** Math ArcCos The Arc Cosine **of** a number, returned **in** radians  
 28009: **Function** Math ArcSin The Arc Sine **of** a number, returned **in** radians  
 28010: **Function** System ArcTan The Arc Tangent **of** a number, returned **in** radians  
 28011: Keyword **Array** A data **type** holding indexable collections **of** data  
 28012: Keyword **As** Used **for** casting **object** references  
 28013: **Procedure** System Assign Assigns a **file** handle **to** a binary **or** text **file**  
 28014: **Function** System Assigned Returns true **if** a reference **is** not **nil**  
 28015: **Procedure** System AssignFile Assigns a **file** handle **to** a binary **or** text **file**  
 28016: **Procedure** Printers AssignPrn Treats the printer **as** a text **file** - an easy way **of** printing text  
 28017:  
 28018: B  
 28019: Compiler Directive \$B Whether **to** short cut **and** **and** **or** operations  
 28020: Compiler Directive \$BoolEval Whether **to** short cut **and** **and** **or** operations  
 28021: **Procedure** SysUtils Beep Make a beep sound  
 28022: Keyword **Begin** Keyword that starts a statement block  
 28023: **Function** System BeginThread Begins a separate thread **of** code execution  
 28024: **Procedure** System BlockRead Reads a block **of** data records from an untyped binary **file**  
 28025: **Procedure** System BlockWrite Writes a block **of** data records **to** an untyped binary **file**  
 28026: **Type** System Boolean Allows just True **and** False values  
 28027: **Function** Classes Bounds Create a TRect value from top left **and** size values  
 28028: **Procedure** System Break Forces a jump **out** **of** a single loop  
 28029: **Type** System Byte An integer **type** supporting values 0 **to** 255  
 28030:  
 28031: C  
 28032: **Type** System Cardinal The basic unsigned integer **type**  
 28033: Keyword **Case** A mechanism **for** acting upon different values **of** an Ordinal  
 28034: **Function** StdConv CelsiusToFahrenheit Convert a celsius temperature into fahrenheit  
 28035: **Function** SysUtils ChangeFileExt Change the extension part **of** a **file** name  
 28036: **Type** System Char Variable **type** holding a single character  
 28037: **Procedure** System ChDir Change the working drive plus path **for** a specified drive  
 28038: **Function** System Chr Convert an integer into a character  
 28039: Keyword **Class** Starts the declaration **of** a **type** **of** object **class**  
 28040: **Procedure** System Close Closes an open **file**  
 28041: **Procedure** System CloseFile Closes an open **file**  
 28042: Variable System CmdLine Holds the execution text used **to** start the current **program**  
 28043: **Type** System Comp A 64 bit signed integer  
 28044: **Function** SysUtils CompareStr Compare two strings **to** see which **is** greater than the other  
 28045: **Function** SysUtils CompareText Compare two strings **for** equality, ignoring **case**  
 28046: **Function** Math CompareValue Compare numeric values **with** a tolerance  
 28047: **Function** System Concat Concatenates one **or** more strings into one **string**  
 28048: Keyword **Const** Starts the definition **of** fixed data values  
 28049: Keyword **Constructor** Defines the method used **to** create an **object** **from** a **class**  
 28050: **Procedure** System Continue Forces a jump **to** the next iteration **of** a loop  
 28051: **Function** ConvUtils Convert Convert one measurement value **to** another  
 28052: **Function** System Copy Create a copy **of** part **of** a **string** **or** an **array**  
 28053: **Function** System Cos The Cosine **of** a number  
 28054: **Function** SysUtils CreateDir Create a directory  
 28055: **Type** System Currency A floating point **type** **with** 4 decimals used **for** financial values  
 28056: Variable SysUtils CurrencyDecimals Defines decimal digit count **in** the Format **function**  
 28057: Variable SysUtils CurrencyFormat Defines currency **string** placement **in** curr display functions  
 28058: Variable SysUtils CurrencyString The currency **string** used **in** currency display functions  
 28059: **Function** SysUtils CurrToStr Convert a currency value **to** a **string**  
 28060: **Function** SysUtils CurrToStrF Convert a currency value **to** a **string** **with** formatting  
 28061:  
 28062: D  
 28063: Compiler Directive \$D Determines whether application debug information **is** built  
 28064: Compiler Directive \$DebugInfo Determines whether application debug information **is** built  
 28065: Compiler Directive \$Define Defines a compiler directive symbol - **as** used by IfDef  
 28066: Compiler Directive \$DefinitionInfo Determines whether application symbol information **is** built  
 28067: **Function** SysUtils Date Gives the current date  
 28068: Variable SysUtils DateSeparator The character used **to** separate display date fields  
 28069: **Function** SysUtils DateTimeToFileDate Convert a TDateTime value **to** a **File** date/time format  
 28070: **Function** SysUtils DateTimeToStr Converts TDateTime date **and** time values **to** a **string**  
 28071: **Procedure** SysUtils DateTimeToString Rich formatting **of** a TDateTime variable **into** a **string**

```

28072: Function SysUtils DateToStr Converts a TDateTime date value to a string
28073: Function DateUtils DayOfMonth Gives day of month index for a TDateTime value (ISO 8601)
28074: Function DateUtils DayOfWeek Gives day of week index for a TDateTime value (ISO 8601)
28075: Function DateUtils DayOfYear Gives the day of the year for a TDateTime value (ISO 8601)
28076: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
28077: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
28078: Function DateUtils DaysInAMonth Gives the number of days in a month
28079: Function DateUtils DaysInAYear Gives the number of days in a year
28080: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
28081: Procedure System Dec Decrement an ordinal variable
28082: Variable SysUtils DecimalSeparator The character used to display the decimal point
28083: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
28084: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
28085: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
28086: Directive Default Defines default processing for a property
28087: Function Math DegToRad Convert a degrees value to radians
28088: Procedure System Delete Delete a section of characters from a string
28089: Function SysUtils DeleteFile Delete a file specified by its file name
28090: Keyword Destructor Defines the method used to destroy an object
28091: Function SysUtils DirectoryExists Returns true if the given directory exists
28092: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
28093: Function SysUtils DiskSize Gives the size in bytes of a specified drive
28094: Procedure System Dispose Dispose of storage used by a pointer type variable
28095: Keyword Div Performs integer division, discarding the remainder
28096: Keyword Do Defines the start of some controlled action
28097: Type System Double A floating point type supporting about 15 digits of precision
28098: Keyword DownTo Prefixes an decremental for loop target value
28099: Function StrUtils DupeString Creates a string containing copies of a substring
28100: Directive Dynamic Allows a class method to be overriden in derived classes
28101:
28102: E
28103: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
28104: Compiler Directive $Endif Terminates conditional code compilation
28105: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
28106: Keyword Else Starts false section of if, case and try statements
28107: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
28108: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
28109: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
28110: Keyword End Keyword that terminates statement blocks
28111: Function DateUtils EndOfDay Generate a TDateTime value set to the very end of a day
28112: Function DateUtils EndOfMonth Generate a TDateTime value set to the very end of a month
28113: Procedure System EndThread Terminates a thread with an exit code
28114: Function System EOF Returns true if a file opened with Reset is at the end
28115: Function System Eoln Returns true if the current text file is pointing at a line end
28116: Procedure System Erase Erase a file
28117: Variable System ErrorAddr Sets the error address when an application terminates
28118: Keyword Except Starts the error trapping clause of a Try statement
28119: Procedure System Exclude Exclude a value in a set variable
28120: Procedure System Exit Exit abruptly from a function or procedure
28121: Variable System ExitCode Sets the return code when an application terminates
28122: Function System Exp Gives the exponent of a number
28123: Directive System Export Makes a function or procedure in a DLL externally available
28124: Type System Extended The floating point type with the highest capacity and precision
28125: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
28126: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
28127: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
28128: Function SysUtils ExtractFileName Extracts the name part of a full file name
28129: Function SysUtils ExtractFilePath Extracts the path part of a full file name
28130:
28131: F
28132: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
28133: Keyword File Defines a typed or untyped file
28134: Function SysUtils FileAge Get the last modified date/time of a file without opening it
28135: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
28136: Function SysUtils FileExists Returns true if the given file exists
28137: Function SysUtils FileGetAttr Gets the attributes of a file
28138: Variable System FileMode Defines how Reset opens a binary file
28139: Function System FilePos Gives the file position in a binary or text file
28140: Function SysUtils FileSearch Search for a file in one or more directories
28141: Function SysUtils FileSetAttr Sets the attributes of a file
28142: Function SysUtils FileSetDate Set the last modified date and time of a file
28143: Function System FileInfo Gives the size in records of an open file
28144: Procedure System FillChar Fills out a section of storage with a fill character or byte value
28145: Keyword Finally Starts the unconditional code section of a Try statement
28146: Function SysUtils FindClose Closes a successful FindFirst file search
28147: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
28148: Function SysUtils FindFirst Finds all files matching a file mask and attributes
28149: Function SysUtils FindNext Find the next file after a successful FindFirst
28150: Function SysUtils FloatToStr Convert a floating point value to a string
28151: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
28152: Procedure System Flush Flushes buffered text file data to the file
28153: Keyword For Starts a loop that executes a finite number of times
28154: Function SysUtils ForceDirectories Create a new path of directories
28155: Function SysUtils Format Rich formatting of numbers and text into a string
28156: Function SysUtils FormatCurr Rich formatting of a currency value into a string
28157: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
28158: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
28159: Function System Frac The fractional part of a floating point number
28160: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil

```

28161: **Procedure** System FreeMem Free memory storage used by a variable  
 28162: Keyword System **Function** Defines a subroutine that returns a value  
 28163:  
 28164: G  
 28165: **Function** SysUtils GetCurrentDir Get the current directory (drive plus directory)  
 28166: **Procedure** System GetDir Get the **default** directory (drive plus path) **for** a specified drive  
 28167: **Function** System GetLastError Gives the error code **of** the last failing Windows API call  
 28168: **Procedure** SysUtils GetLocaleFormatSettings Gets locale values **for** thread-safe functions  
 28169: **Function** System GetMem Get a specified number **of** storage bytes  
 28170: Keyword **Goto** Forces a jump **to** a **label**, regardless **of** nesting  
 28171:  
 28172: H  
 28173: Compiler Directive \$H Treat **string** types **as** AnsiString **or** ShortString  
 28174: Compiler Directive \$Hints Determines whether Delphi shows compilation hints  
 28175: **Procedure** System Halt Terminates the **program** **with** an optional dialog  
 28176: **Function** System Hi Returns the hi-order byte **of** a (2 byte) Integer  
 28177: **Function** System High Returns the highest value **of** a **type** **or** variable  
 28178:  
 28179: I  
 28180: Compiler Directive \$I Allows code **in** an include **file** **to** be incorporated into a **Unit**  
 28181: Compiler Directive \$IfDef Executes code **if** a conditional symbol has been defined  
 28182: Compiler Directive \$IfNDef Executes code **if** a conditional symbol has **not** been defined  
 28183: Compiler Directive \$IfOpt Tests **for** the state **of** a Compiler directive  
 28184: Compiler Directive \$Include Allows code **in** an include **file** **to** be incorporated into a **Unit**  
 28185: Compiler Directive \$IOChecks When **on**, an IO operation error throws an exception  
 28186: Keyword **If** Starts a conditional expression **to** determine what **to do** next  
 28187: Keyword **Implementation** Starts the **implementation** (code) section **of** a **Unit**  
 28188: Keyword **In** Used to test **if** a value **is** a member **of** a **set**  
 28189: **Procedure** System Inc Increment an ordinal variable  
 28190: **Function** DateUtils IncDay Increments a TDateTime variable by + **or** - number **of** days  
 28191: **Procedure** System Include Include a value **in** a **set** variable  
 28192: **Function** DateUtils IncMillisecond Increments a TDateTime variable by + **or** - number **of** milliseconds  
 28193: **Function** DateUtils IncMinute Increments a TDateTime variable by + **or** - number **of** minutes  
 28194: **Function** SysUtils IncMonth Increments a TDateTime variable by a number **of** months  
 28195: **Function** DateUtils IncSecond Increments a TDateTime variable by + **or** - number **of** seconds  
 28196: **Function** DateUtils IncYear Increments a TDateTime variable by a number **of** years  
 28197: Directive Index Principally defines indexed **class** data properties  
 28198: Constant Math Infinity Floating point value **of** infinite size  
 28199: Keyword **Inherited** Used to call the parent **class** **constructor** **or** **destructor** method  
 28200: Variable System Input Defines the standard input text **file**  
 28201: **Function** Dialogs InputBox Display a dialog that asks **for** user text input, **with default**  
 28202: **Function** Dialogs InputQuery Display a dialog that asks **for** user text input  
 28203: **Procedure** System Insert Insert a **string** **into** another **string**  
 28204: **Function** System Int The integer part **of** a floating point number **as** a float  
 28205: **Type** System Int64 A 64 bit sized integer - the largest **in** Delphi  
 28206: **Type** System Integer The basic Integer **type**  
 28207: Keyword System **Interface** Used **for** Unit external definitions, **and as** a Class skeleton  
 28208: **Function** SysUtils IntToHex Convert an Integer into a hexadecimal **string**  
 28209: **Function** SysUtils IntToStr Convert an integer into a **string**  
 28210: **Function** System IOResult Holds the return code **of** the last I/O operation  
 28211: Keyword **Is** Tests whether an **object** **is** a certain **class** **or** ascendant  
 28212: **Function** Math IsInfinite Checks whether a floating point number **is** infinite  
 28213: **Function** SysUtils IsLeapYear Returns true **if** a given calendar year **is** a leap year  
 28214: **Function** System IsMultiThread Returns true **if** the code **is** running multiple threads  
 28215: **Function** Math IsNaN Checks **to see if** a floating point number holds a real number  
 28216:  
 28217: L  
 28218: Compiler Directive \$L Determines what application debug information **is** built  
 28219: Compiler Directive \$LocalSymbols Determines what application debug information **is** built  
 28220: Compiler Directive \$LongStrings Treat **string** types **as** AnsiString **or** ShortString  
 28221: **Function** SysUtils LastDelimiter Find the last position **of** selected characters **in** a **string**  
 28222: **Function** System Length Return the number **of** elements **in** an **array** **or** **string**  
 28223: **Function** System Ln Gives the natural logarithm **of** a number  
 28224: **Function** System Lo Returns the low-order byte **of** a (2 byte) Integer  
 28225: **Function** Math Log10 Gives the log **to base** 10 **of** a number  
 28226: Variable SysUtils LongDateFormat Long version **of** the date **to string** format  
 28227: Variable SysUtils LongDayNames An **array** **of** days **of** the week names, starting 1 = Sunday  
 28228: **Type** System LongInt An Integer whose size **is** guaranteed **to be** 32 bits  
 28229: Variable SysUtils LongMonthNames An **array** **of** days **of** the month names, starting 1 = January  
 28230: Variable SysUtils LongTimeFormat Long version **of** the time **to string** format  
 28231: **Type** System LongWord A 32 bit unsigned integer  
 28232: **Function** System Low Returns the lowest value **of** a **type** **or** variable  
 28233: **Function** SysUtils LowerCase Change upper case characters **in** a **string** **to lower case**  
 28234:  
 28235: M  
 28236: Compiler Directive \$MinEnumSize Sets the minimum storage used **to hold** enumerated types  
 28237: **Function** Math Max Gives the maximum **of** two integer values  
 28238: Constant System MaxInt The maximum value an Integer can have  
 28239: Constant System MaxLongInt The maximum value an LongInt can have  
 28240: **Function** Math Mean Gives the average **for** a **set of** numbers  
 28241: **Function** Dialogs MessageDlg Displays a **message**, **symbol**, **and** selectable buttons  
 28242: **Function** Dialogs MessageDlgPos Displays a **message** plus buttons at a given screen position  
 28243: **Function** Math Min Gives the minimum **of** two integer values  
 28244: Constant SysUtils MinsPerDay Gives the number **of** minutes **in** a day  
 28245: **Procedure** System MkDir Make a directory  
 28246: Keyword **Mod** Performs integer division, returning the remainder  
 28247: Constant SysUtils MonthDays Gives the number **of** days **in** a month  
 28248: **Function** DateUtils MonthOfTheYear Gives the month **of** the year **for** a TDateTime value  
 28249: **Procedure** System Move Copy bytes **of** data from a source **to** a destination

28250:  
 28251: N  
 28252: Constant Math NaN **Not** a real number  
 28253: Variable SysUtils NegCurrFormat Defines negative amount formatting **in** currency displays  
 28254: **Procedure** System New Create a new pointer **type** variable  
 28255: Constant System **Nil** A pointer value that **is** defined **as** undetermined  
 28256: Keyword **Not** Boolean **Not** or bitwise **not of** one arguments  
 28257: **Function** SysUtils Now Gives the current date **and** time  
 28258: Variable Variants Null A variable that has no value  
 28259:  
 28260: O  
 28261: Compiler Directive \$O Determines whether Delphi optimises code when compiling  
 28262: Compiler Directive \$Optimization Determines whether Delphi optimises code when compiling  
 28263: Compiler Directive \$OverFlowChecks Determines whether Delphi checks integer **and** enum bounds  
 28264: Keyword System **Object** Allows a subroutine data **type** to refer **to** an **object** method  
 28265: **Function** System Odd Tests whether an integer has an odd value  
 28266: Keyword **Of** Linking keyword used **in** many places  
 28267: Keyword **On** Defines exception handling **in** a **Try Except** clause  
 28268: Keyword **Or** Boolean **or** or bitwise **or of** two arguments  
 28269: **Function** System Ord Provides the Ordinal value **of** an integer, character **or** enum  
 28270: Directive **Out** Identifies a routine parameter **for** output only  
 28271: Variable System Output Defines the standard output text **file**  
 28272: Directive **Overload** Allows **2 or** more routines **to** have the same name  
 28273: Directive **Override** Defines a method that replaces a **virtual** parent **class** method  
 28274:  
 28275: P  
 28276: Keyword **Packed** Compacts complex data types into minimal storage  
 28277: **Type** System PAnsiChar A pointer **to** an AnsiChar value  
 28278: **Type** System PAnsiString Pointer **to** an AnsiString value  
 28279: **Function** System ParamCount Gives the number **of** parameters passed **to** the current **program**  
 28280: **Function** System ParamStr Returns one **of** the parameters used **to** run the current **program**  
 28281: **Type** System PChar A pointer **to** an Char value  
 28282: **Type** System PCurrency Pointer **to** a Currency value  
 28283: **Type** System PDateTime Pointer **to** a TDateTime value  
 28284: **Type** System PEextended Pointer **to** a Extended floating point value  
 28285: **Function** System Pi The mathematical constant  
 28286: **Type** System PInt64 Pointer **to** an Int64 value  
 28287: **Function** Classes Point Generates a TPoint value from X **and** Y values  
 28288: **Type** System Pointer Defines a general use Pointer **to** any memory based data  
 28289: **Function** Classes PointsEqual Compares two TPoint values **for** equality  
 28290: **Function** System Pos Find the position **of** one **string** in another  
 28291: **Function** System Pred Decrement an ordinal variable  
 28292: **Function** Printers Printer Returns a reference **to** the global Printer **object**  
 28293: Directive **Private** Starts the section **of** **private** data **and** methods **in** a **class**  
 28294: Keyword System **Procedure** Defines a subroutine that does **not** return a value  
 28295: **Procedure** FileCtrl ProcessPath Split a drive/path/filename **string** into its constituent parts  
 28296: Keyword System **Program** Defines the start **of** an application  
 28297: **Function** Dialogs PromptForFileName Shows a dialog allowing the user **to** select a **file**  
 28298: Keyword System **Property** Defines controlled access **to** **class** fields  
 28299: Directive **Protected** Starts a section **of** **class private** data accesible **to** sub-classes  
 28300: **Type** System PShortString A pointer **to** an ShortString value  
 28301: **Type** System PString Pointer **to** a String value  
 28302: **Function** Types PtInRect Tests to see if a point lies within a rectangle  
 28303: Directive **Public** Starts an externally accessible section **of** a **class**  
 28304: Directive **Published** Starts a published externally accessible section **of** a **class**  
 28305: **Type** System PVariant Pointer **to** a Variant value  
 28306: **Type** System PWideChar Pointer **to** a WideChar  
 28307: **Type** System PWideString Pointer **to** a WideString value  
 28308:  
 28309: Q  
 28310: Compiler Directive \$Q Determines whether Delphi checks integer **and** enum bounds  
 28311:  
 28312: R  
 28313: Compiler Directive \$R Determines whether Delphi checks **array** bounds  
 28314: Compiler Directive \$RangeChecks Determines whether Delphi checks **array** bounds  
 28315: Compiler Directive \$ReferenceInfo Determines whether symbol reference information **is** built  
 28316: Compiler Directive \$Resource Defines a resource **file** **to** be included **in** the application linking  
 28317: **Function** Math RadToDeg Converts a radian value **to** degrees  
 28318: Keyword **Raise** Raise an exception  
 28319: **Function** System Random Generate a random floating point **or** integer number  
 28320: **Procedure** System Randomize Reposition the Random number generator next value  
 28321: **Function** Math RandomRange Generate a random integer number within a supplied range  
 28322: Variable System RandSeed Reposition the Random number generator next value  
 28323: **Procedure** System **Read** Read data from a binary **or** text **file**  
 28324: **Procedure** System ReadLn Read a complete line **of** data from a text **file**  
 28325: **Type** System Real A floating point **type** supporting about **15** digits **of** precision  
 28326: **Type** System Real48 The floating point **type** with the highest capacity **and** precision  
 28327: **Procedure** System ReallocMem Reallocate an existing block **of** storage  
 28328: **Function** DateUtils RecodeDate Change only the date part **of** a TDateTime variable  
 28329: **Function** DateUtils RecodeTime Change only the time part **of** a TDateTime variable  
 28330: Keyword **Record** A structured data **type** - holding fields **of** data  
 28331: **Function** Classes Rect Create a TRect value from **2** points **or** **4** coordinates  
 28332: **Function** SysUtils RemoveDir Remove a directory  
 28333: **Procedure** System Rename Rename a **file**  
 28334: **Function** SysUtils RenameFile Rename a **file** **or** directory  
 28335: Keyword **Repeat** Repeat statements **until** a ternmination condition **is** met  
 28336: **Procedure** SysUtils ReplaceDate Change only the date part **of** a TDateTime variable  
 28337: **Procedure** SysUtils ReplaceTime Change only the time part **of** a TDateTime variable  
 28338: **Procedure** System Reset Open a text **file** **for** reading, **or** binary **file** **for** read/write

28339: Variable System Result A variable used **to** hold the return value from a **function**  
 28340: **Procedure** System ReWrite Open a text **or** binary **file for write** access  
 28341: **Procedure** System RmDir Remove a directory  
 28342: **Function** System Round Rounds a floating point number **to** an integer  
 28343: **Procedure** System RunError Terminates the **program with** an error dialog  
 28344:  
 28345: S  
 28346: Constant SysUtils SecsPerDay Gives the number **of** seconds **in** a day  
 28347: **Procedure** System Seek Move the pointer **in** a binary **file to** a new **record** position  
 28348: **Function** System SeekEof Skip **to** the **end of** the current line **or file**  
 28349: **Function** System SeekEoln Skip **to** the **end of** the current line **or file**  
 28350: **Function** FileCtrl SelectDirectory Display a dialog **to** allow user selection **of** a directory  
 28351: Variable System Self Hidden parameter **to** a method - refers **to** the containing **object**  
 28352: Keyword Set Defines a **set of** up **to** 255 distinct values  
 28353: **Function** SysUtils SetCurrentDir Change the current directory  
 28354: **Procedure** System SetLength Changes the size **of** a **string, or the size(s) of** an **array**  
 28355: **Procedure** System SetString Copies characters from a buffer into a **string**  
 28356: Keyword Shl Shift an integer value left by a number **of** bits  
 28357: Variable SysUtils ShortDateFormat Compact version **of** the date **to string** format  
 28358: Variable SysUtils ShortDayNames An **array of** days **of** the week names, starting 1 = Sunday  
 28359: **Type** System ShortInt An integer **type** supporting values -128 to 127  
 28360: Variable SysUtils ShortMonthNames An **array of** days **of** the month names, starting 1 = Jan  
 28361: **Type** System ShortString Defines a **string of** up **to** 255 characters  
 28362: Variable SysUtils ShortTimeFormat Short version **of** the time **to string** format  
 28363: **Procedure** Dialogs ShowMessage Display a **string** in a simple dialog **with** an OK button  
 28364: **Procedure** Dialogs ShowMessageFmt Display formatted data **in** a simple dialog **with** an OK button  
 28365: **Procedure** Dialogs ShowMessagePos Display a **string** **in** a simple dialog at a given screen position  
 28366: Keyword Shr Shift an integer value right by a number **of** bits  
 28367: **Function** System Sin The Sine **of** a number  
 28368: **Type** System Single The smallest capacity **and** precision floating point **type**  
 28369: **Function** System SizeOf Gives the storage byte size **of** a **type or** variable  
 28370: **Function** System Slice Creates a slice **of** an **array as** an Open **Array** parameter  
 28371: **Type** System SmallInt An Integer **type** supporting values from -32768 to 32767  
 28372: **Function** System Sqr Gives the square **of** a number  
 28373: **Function** System Sqrt Gives the square root **of** a number  
 28374: **Procedure** System Str Converts an integer **or** floating point number **to a string**  
 28375: **Type** System String A data **type** that holds a **string of** characters  
 28376: **Function** System StringOfChar Creates a **string with** one character repeated many times  
 28377: **Function** SysUtils StringReplace Replace one **or** more substrings found within a **string**  
 28378: **Function** System StringToWideChar Converts a normal **string** **into** a WideChar 0 terminated buffer  
 28379: **Function** SysUtils StrScan Searches **for** a specific character **in** a constant **string**  
 28380: **Function** SysUtils StrToCurr Convert a number **string** **into** a currency value  
 28381: **Function** SysUtils StrToDate Converts a date **string** **into** a TDateTime value  
 28382: **Function** SysUtils StrToDateTm Converts a date+time **string** **into** a TDateTime value  
 28383: **Function** SysUtils StrToFloat Convert a number **string** **into** a floating point value  
 28384: **Function** SysUtils StrToInt Convert an integer **string** **into** an Integer value  
 28385: **Function** SysUtils StrToInt64 Convert an integer **string** **into** an Int64 value  
 28386: **Function** SysUtils StrToInt64Def Convert a **string** **into** an Int64 value **with default**  
 28387: **Function** SysUtils StrToIntDef Convert a **string** **into** an Integer value **with default**  
 28388: **Function** SysUtils StrToTime Converts a time **string** **into** a TDateTime value  
 28389: **Function** StrUtils StuffString Replaces a part **of** one **string with** another  
 28390: **Function** System Succ Increment an ordinal variable  
 28391: **Function** Math Sum Return the sum **of** an **array of** floating point values  
 28392:  
 28393: T  
 28394: **Function** Math Tan The Tangent **of** a number  
 28395: **Type** Classes TBits An **object** that can hold an infinite number **of** Boolean values  
 28396: Variable ConvUtils TConvFamily Defines a family **of** measurement types **as** used by Convert  
 28397: **Type** ConvUtils TConvType Defines a measurement **type as** used by Convert  
 28398: **Type** System TDateTime Data **type** holding a date **and** time value  
 28399: **Type** System Text Defines a **file as** a text **file**  
 28400: **Type** System TextFile Declares a **file type** for storing lines **of** text  
 28401: **Type** SysUtils TFloatFormat Formats **for** use **in** floating point number display functions  
 28402: **Type** SysUtils TFormatSettings A **record for** holding locale values **for** thread-safe functions  
 28403: Keyword Then Part **of** an if statement - starts the true clause  
 28404: Variable SysUtils ThousandSeparator The character used **to** display the thousands separator  
 28405: Keyword ThreadVar Defines variables that are given separate instances per thread  
 28406: **Function** SysUtils Time Gives the current time  
 28407: Variable SysUtils TimeAMString Determines AM value **in** DateTimeToString **procedure**  
 28408: Variable SysUtils TimePMString Determines PM value **in** DateTimeToString **procedure**  
 28409: Variable SysUtils TimeSeparator The character used **to** separate display time fields  
 28410: **Function** SysUtils TimeToStr Converts a TDateTime time value **to a string**  
 28411: **Type** Classes TList General purpose container **of** a list **of** objects  
 28412: Keyword To Prefixes an incremental **for** loop target value  
 28413: **Type** System TObject The base **class type** that **is** ancestor **to** all other classes  
 28414: **Function** DateUtils Tomorrow Gives the date tomorrow  
 28415: **Type** Dialogs TOpenDialog Displays a **file** selection dialog  
 28416: **Type** Types TPoint Holds X **and** Y integer values  
 28417: **Type** Dialogs TPrintDialog **Class** that creates a printer selection **and** control dialog  
 28418: **Type** Types TRect Holds rectangle coordinate values  
 28419: **Type** SysUtils TReplaceFlags Defines options **for** the StringReplace routine  
 28420: **Function** SysUtils Trim Removes leading **and** trailing blanks from a **string**  
 28421: **Function** SysUtils TrimLeft Removes leading blanks from a **string**  
 28422: **Function** SysUtils TrimRight Removes trailing blanks from a **string**  
 28423: **Function** System Trunc The integer part **of** a floating point number  
 28424: **Procedure** System Truncate Truncates a **file size** - removes all data after the current position  
 28425: Keyword Try Starts code that has error trapping  
 28426: **Type** Dialogs TSaveDialog Displays a dialog **for** selecting a save **file name**  
 28427: **Type** SysUtils TSearchRec **Record** used **to** hold data **for** FindFirst **and** FindNext

```

28428: Type Classes TStringList Holds a variable length list of strings
28429: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
28430: Type System TThreadFunc Defines the function to be called by BeginThread
28431: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
28432: Keyword Type Defines a new category of variable or process
28433:
28434: U
28435: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
28436: Keyword Unit Defines the start of a unit file - a Delphi module
28437: Keyword Until Ends a Repeat control loop
28438: Function System UpCase Convert a Char value to upper case
28439: Function SysUtils UpperCase Change lower case characters in a string to upper case
28440: Keyword Uses Declares a list of Units to be imported
28441:
28442: V
28443: Procedure System Val Converts number strings to integer and floating point values
28444: Keyword Var Starts the definition of a section of data variables
28445: Type System Variant A variable type that can hold changing data types
28446: Function Variants VarType Gives the current type of a Variant variable
28447: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
28448: Directive Virtual Allows a class method to be overriden in derived classes
28449:
28450: W
28451: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
28452: Keyword While Repeat statements whilst a continuation condition is met
28453: Type System WideChar Variable type holding a single International character
28454: Function System WideStringToString Copies a null terminated WideChar string to a normal string
28455: Type System WideString A data type that holds a string of WideChars
28456: Keyword With A means of simplifying references to structured variables
28457: Type System Word An integer type supporting values 0 to 65535
28458: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
28459: Procedure System Write Write data to a binary or text file
28460: Procedure System WriteLn Write a complete line of data to a text file
28461:
28462: X
28463: Compiler Directive $X Controls some Pascal extension handling
28464: Keyword Xor Boolean Xor or bitwise Xor of two arguments
28465: Y
28466: Compiler Directive $Y Determines whether application symbol information is built
28467: Function DateUtils Yesterday Gives the date yesterday
28468:
28469: Z
28470: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
28471:
28472:
28473:
28474: procedure SIRegister_uPSUtils(CL: TPSPascalCompiler);
28475: begin //'TPSBaseType', '').SetString( Byte);
28476: PSMainProcName', 'String').SetString( '!MAIN');
28477: PSMainProcNameOrg', 'String').SetString( 'Main Proc');
28478: 'PSLowBuildSupport', 'LongInt').SetInt( 12);
28479: 'PSCurrentBuildNo', 'LongInt').SetInt( 23);
28480: 'PSCurrentversion', 'String').SetString( '1.31');
28481: 'PSValidHeader', 'LongInt').SetInt( 1397769801);
28482: 'PSAddrStackStart', 'LongInt').SetInt( 1610612736);
28483: 'PSAddrNegativeStackStart', 'LongInt').SetInt( 1073741824);
28484: 'btReturnAddress', 'LongInt').SetInt( 0);
28485: 'btU8', 'LongInt').SetInt( 1);
28486: 'btS8', 'LongInt').SetInt( 2);
28487: 'btU16', 'LongInt').SetInt( 3);
28488: 'btS16', 'LongInt').SetInt( 4);
28489: 'btU32', 'LongInt').SetInt( 5);
28490: 'btS32', 'LongInt').SetInt( 6);
28491: 'btSingle', 'LongInt').SetInt( 7);
28492: 'btDouble', 'LongInt').SetInt( 8);
28493: 'btExtended', 'LongInt').SetInt( 9);
28494: 'btString', 'LongInt').SetInt( 10);
28495: 'btRecord', 'LongInt').SetInt( 11);
28496: 'btArray', 'LongInt').SetInt( 12);
28497: 'btPointer', 'LongInt').SetInt( 13);
28498: 'btPChar', 'LongInt').SetInt( 14);
28499: 'btResourcePointer', 'LongInt').SetInt( 15);
28500: 'btVariant', 'LongInt').SetInt( 16);
28501: 'btS64', 'LongInt').SetInt( 17);
28502: 'btU64', 'LongInt').SetInt( 30);
28503: 'btChar', 'LongInt').SetInt( 18);
28504: 'btWideString', 'LongInt').SetInt( 19);
28505: 'btWideChar', 'LongInt').SetInt( 20);
28506: 'btProcPtr', 'LongInt').SetInt( 21);
28507: 'btStaticArray', 'LongInt').SetInt( 22);
28508: 'btSet', 'LongInt').SetInt( 23);
28509: 'btCurrency', 'LongInt').SetInt( 24);
28510: 'btClass', 'LongInt').SetInt( 25);
28511: 'btInterface', 'LongInt').SetInt( 26);
28512: 'btNotificationVariant', 'LongInt').SetInt( 27);
28513: 'btUnicodeString', 'LongInt').SetInt( 28);
28514: 'btType', 'LongInt').SetInt( 130);
28515: 'btEnum', 'LongInt').SetInt( 129);
28516: 'btExtClass', 'LongInt').SetInt( 131);

```

```

28517: 'CM_A','LongInt').SetInt( 0);
28518: 'CM_CA','LongInt').SetInt( 1);
28519: 'CM_P','LongInt').SetInt( 2);
28520: 'CM_PV','LongInt').SetInt( 3);
28521: 'CM_PO','LongInt').SetInt( 4);
28522: 'Cm_C','LongInt').SetInt( 5);
28523: 'Cm_G','LongInt').SetInt( 6);
28524: 'Cm_CG','LongInt').SetInt( 7);
28525: 'Cm_CNG','LongInt').SetInt( 8);
28526: 'Cm_R','LongInt').SetInt( 9);
28527: 'Cm_ST','LongInt').SetInt( 10);
28528: 'Cm_Pt','LongInt').SetInt( 11);
28529: 'CM_CO','LongInt').SetInt( 12);
28530: 'Cm_cv','LongInt').SetInt( 13);
28531: 'cm_sp','LongInt').SetInt( 14);
28532: 'cm_bn','LongInt').SetInt( 15);
28533: 'cm_vm','LongInt').SetInt( 16);
28534: 'cm_sf','LongInt').SetInt( 17);
28535: 'cm_fg','LongInt').SetInt( 18);
28536: 'cm_puevh','LongInt').SetInt( 19);
28537: 'cm_poexh','LongInt').SetInt( 20);
28538: 'cm_in','LongInt').SetInt( 21);
28539: 'cm_spc','LongInt').SetInt( 22);
28540: 'cm_inc','LongInt').SetInt( 23);
28541: 'cm_dec','LongInt').SetInt( 24);
28542: 'cm_nop','LongInt').SetInt( 255);
28543: 'Cm_PG','LongInt').SetInt( 25);
28544: 'Cm_P2G','LongInt').SetInt( 26);
28545: CL.AddTypeS('TbtU8', 'Byte');
28546: CL.AddTypeS('TbtS8', 'ShortInt');
28547: CL.AddTypeS('TbtU16', 'Word');
28548: CL.AddTypeS('TbtS16', 'SmallInt');
28549: CL.AddTypeS('TbtU32', 'Cardinal');
28550: CL.AddTypeS('Tbts32', 'Longint');
28551: CL.AddTypeS('TbtSingle', 'Single');
28552: CL.AddTypeS('TbtDouble', 'double');
28553: CL.AddTypeS('TbtExtended', 'Extended');
28554: CL.AddTypeS('tbtCurrency', 'Currency');
28555: CL.AddTypeS('tbts64', 'int64');
28556: CL.AddTypeS('Tbtu64', 'uint64');
28557: CL.AddTypeS('TbtString', 'string');
28558: Function MakeHash( const s : TbtString ) : Longint;
28559: // TbtString = {$IFDEF DELPHI2009UP}AnsiString{$ELSE}String{$ENDIF};
28560: // 'PointerSize','LongInt').SetInt( IPointer ( 8 4 ));
28561: end;
28562:
28563: -----
28564: mapX:
28565: if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
28566:   writeln('cologne map found');
28567:   GetGeoMAP('html',ExePath+AFilename2,'dom cologne')
28568:   writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
28569:   OpenMapX('church trier');
28570:   GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
28571:   writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));
28572:   >>> //latitude: '50.94133705' longitude: '6.95812076100766'
28573:   // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
28574:   writeln(GetGeoCoord('xml','church cefalu sicily',true));
28575:   CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
28576:   Function SendInput( cInputs : UINT; var pInputs : TInput; cbSize : Integer) : UINT';
28577:   Function GetLastInputInfo( var plii : TLastInputInfo ) : BOOL';
28578:   Procedure JvErrorIntercept';
28579:   writeln(GetGeoInfo4('178.196.192.131', UrlGeoLookupInfo3));
28580:
28581: {$IFDEF MSWINDOWS}
28582: procedure Process;
28583: var
28584:   Msg: TMsg;
28585: begin
28586:   if ApplicationHasPriority then begin
28587:     Application.ProcessMessages;
28588:   end else begin
28589:     // This guarantees it will not ever call Application.Idle
28590:     if PeekMessage(Msg, 0, 0, 0, PM_NOREMOVE) then begin
28591:       Application.HandleMessage;
28592:     end;
28593:   end;
28594: end;
28595:
28596:
28597:
28598: maxBox Ref:
28599: Signature:
28600:
28601: SHA1: of maxBox 4.2.0.80 638E7412750AB0ECF14F2A5515BC4A2DE561EAC2
28602: CRC32 of maxBox4.exe: 7C91FD2A
28603: maxBox4.exe 26,650,112 bytes
28604:
28605:

```

```

28606: Ref:
28607:   1. writeln(SHA1(Exepath+'\maxbox4.exe'))
28608:   2. shdig: TSHA1Digest;
28609:     shdig:= GetSHA1OfFile(false,exepath+'\maxbox4.exe');
28610:       for i:= 0 to 19 do write(BytetoHex(shdig[i]));
28611:
28612:   3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox4.exe'),4));
28613:
28614:   4. writeln(shaltohex(SHA1ofStr(loadStringOfFile(exepath+'\maxbox4.exe'))))
28615:
28616: I have to get only the numbers, without the hyphen and the dots, to my calcs worth.
28617:   sr := '123.456.789-00'
28618:   writeln(ReplaceRegExpr ('\D',sr,'',true))
28619:
28620: function GetLocalStream(C_form: string; const lat,long: string):boolean;
28621:   var encodedURL, UrlMapQuestAPI, bufstr: string;
28622:   mStream: TMemoryStream;
28623:   idHTTP: TIdHTTP// THTTPSSend;
28624:   len: integer;
28625:   begin
28626:     UrlMapQuestAPI:= 'http://192.168.1.53:9000';
28627:     encodedURL:= UrlMapQuestAPI;
28628:     idHTTP:= TIdHTTP.Create(NIL)
28629:     mStream:= TMemoryStream.create;
28630:     try
28631:       //HttpGet(EncodedURL, mStream); {WinInet}
28632:       idHTTP.Get1(EncodedURL, mStream)
28633:       mStream.Position:= 0;
28634:       writeln('stream size: '+inttostr(mStream.size));
28635:       //mStream.memory;
28636:       len:= mStream.Size - mStream.Position;
28637:       SetLength(bufstr, len);
28638:       mStream.readbuffer(bufstr, len)
28639:       writeln('debug stream local back: '+bufstr)
28640:     finally
28641:       encodedURL:= '';
28642:       mStream.Free;
28643:       idHTTP.Disconnect
28644:       idHTTP.Free;
28645:       result:= true;
28646:     end;
28647:   end;
28648:
28649:
28650: https://www.virustotal.
com/en/file/74089df9d60f94fd5e6152e69a88f7c35e25ea1084d22f594c490eb7411eaba1/analysis/1454270356/
28651: https://www.virustotal.
com/en/file/cff7c6c0c41256c58d4d5b5ebef958f0df711b9ac219c261f3d65eb42b82673b/analysis/1453638284/
28652: https://www.virustotal.
com/en/file/a6a0a9c798633c3720628669fd384ad9fb2e9a4a6441bb3974b5455e7fdb589c/analysis/1454434882/
28653: https://www.virustotal.
com/en/file/584ca53d6dd8f7de17d0a1959bf78aad04697cf0ad7b3f23aee90b5ff720ede1/analysis/1460194451/
28654:
28655: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3140768)
28656: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3124263).
28657: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3124262).
28658:
28659: VirusTotal metadata old base version mX3
28660: -----
28661: First submission 2015-05-12 23:00:59 UTC ( 22 hours, 2 minutes ago )
28662: Last submission 2015-05-12 23:07:07 UTC ( 21 hours, 56 minutes ago )
28663: File names Surprise.mX4
28664: maXbox3.exe
28665: maxbox3_9.exe
28666:
28667: SHA256: c2ccaafe689c56d4201ad18bbcf2007ec8cb6e5f86d006028aee2709a311b91d
28668: File name: maxbox3.exe
28669: Detection ratio: 0 / 56
28670: Analysis date: 2015-05-12 23:07:07 UTC ( 21 hours, 55 minutes ago )
28671:
28672: MD5 f08ede9627c8434d35e04ff56e529450
28673: SHA1 f0ab7d054111f5ce46ba122d6280397a841c6fab
28674: SHA256 c2ccaafe689c56d4201ad18bbcf2007ec8cb6e5f86d006028aee2709a311b91d
28675: ssdeep 393216:UL1w/LSA+4smc/JQUNDIF31Mr6JmDF32J:1w/LSd/ivZJ
28676: authentihash 0401f343b6a00940ae2f3d2d7c0661729962da3956ab9e339c5610427b1a1d7
28677: imphash 2c4488ddale3c370f9740c2c7f22e2b8
28678: File size 24.0 MB ( 25152000 bytes )
28679: File type Win32 EXE
28680: Magic literal
28681: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
28682:
28683: TrID Windows ActiveX control (36.4%)
28684: Inno Setup installer (34.3%)
28685: InstallShield setup (13.4%)
28686: Win32 EXE PECompact compressed (generic) (13.0%)
28687: Win32 Executable (generic) (1.4%)
28688:
28689: ExifTool file metadata
28690: -----

```

```
28691:  
28692: SpecialBuild mX4  
28693: LegalTrademarks maxbox  
28694: SubsystemVersion 4.0  
28695: Comments reduce to the max  
28696: LinkerVersion 2.25  
28697: ImageVersion 0.0  
28698: FileSubtype 0  
28699: FileVersionNumber 3.9.9.195  
28700: LanguageCode German (Swiss)  
28701: FileFlagsMask 0x003f  
28702: FileDescription maxbox Delphi VM  
28703: CharacterSet Windows, Latin1  
28704: InitializedDataSize 5472768  
28705: FileOS Win32  
28706: Timestamp 2015:05:12 22:08:10+01:00  
28707: MIMEType application/octet-stream  
28708: LegalCopyright Free Pascal Script  
28709: FileVersion 3.9.9.195  
28710: SpeziellesBuild mX4 Compiler Engine  
28711: FileType Win32 EXE  
28712: PEType PE32  
28713: InternalName Surprise mX4  
28714: ProductVersion 3.9 Solar mX4  
28715: UninitializedDataSize 0  
28716: OSVersion 4.0  
28717: OriginalFilename maxbox3_9.exe  
28718: Subsystem Windows GUI  
28719: MachineType Intel 386 or later, and compatibles  
28720: CompanyName kleiner kommunikation  
28721:  
28722: CodeSize 19678208  
28723: ProductName maxbox  
28724: ProductVersionNumber 3.9.9.195  
28725: EntryPoint 0x12c60b0  
28726: ObjectFileType Executable application  
28727:  
28728: ExifTool file metadata  
28729: -----  
28730: Developer metadata  
28731: Publisher kleiner kommunikation  
28732: Product maxbox  
28733: Original name maxbox3_9.exe  
28734: Internal name Surprise mX4  
28735: File version 3.9.9.195  
28736: Description maxbox Delphi VM  
28737: Comments reduce to the max  
28738:  
28739: PE header basic information  
28740:  
28741: Target machine Intel 386 or later processors and compatible processors  
28742: Compilation timestamp 2015-05-12 21:08:10  
28743: Link date 10:08 PM 5/12/2015  
28744: Entry Point 0x012C60B0  
28745: Number of sections 10  
28746:  
28747: 10 PE sections  
28748:  
28749: Name Virtual address Virtual size Raw size Entropy MD5  
28750: .text 4096 19628308 19628544 6.60 3b26cba0ce38b8c057d45ad84384072c  
28751: .itext 19636224 49652 49664 6.60 9141fa3ea9a8a714a3d326cadbe20014  
28752: .data 19689472 333556 333824 5.62 6ca6e99fc0b042f5c3e9a8fffb0f51f1  
28753: .bss 20025344 385568 0 0.00 d41d8cd98f00b204e9800998ecf8427e  
28754: .idata 20414464 58112 58368 5.62 80e9b2e2c89cbad47ec17884f8361a0d  
28755: .edata 20475904 77 512 0.90 ee758732620dcf83f0f65dda98a387b1  
28756: .tbs 20480000 216 0 0.00 d41d8cd98f00b204e9800998ecf8427e  
28757: .rdata 20484096 24 512 0.27 422aa921504e96668acd9676bf246a9f  
28758: .reloc 20488192 1251180 1251328 6.76 29c0e0a70410ce57514a9c3a7ad4903c  
28759: .rsrc 21741568 3828224 3828224 5.38 7f3a42dbc5e62c07f3017151e47123ce  
28760:  
28761: PE exports CreateIncome  
28762:  
28763: Number of PE resources by type  
28764:  
28765: RT_BITMAP 1020  
28766: RT_STRING 317  
28767: RT_RCDATA 181  
28768: RT_ICON 62  
28769: RT_CURSOR 47  
28770: RT_GROUP_ICON 46  
28771: RT_GROUP_CURSOR 41  
28772: WAVE 9  
28773: RT_DIALOG 2  
28774: RT_HTML 2  
28775: RT_MESSAGETABLE 1  
28776: RT_MANIFEST 1  
28777: AVI 1  
28778: RT_VERSION 1  
28779:
```

```

28780:
28781: 38 PE imports
28782:
28783: [+] AVICAP32.DLL   [+] AVICAP32.dll   [+] GLU32.dll
28784: [+] IMAGEHLP.DLL   [+] MSVCRT.DLL   [+] MSVFW32.DLL
28785: [+] OpenGL32.dll   [+] SHFolder.dll   [+] URLMON.DLL
28786: [+] advapi32.dll   [+] comct132.dll   [+] comdlg32.dll
28787: [+] gdi32.dll     [+] imagehlp.dll   [+] imm32.dll
28788: [+] iphpapi.dll   [+] kerne132.dll   [+] mpr.dll
28789: [+] msacm32.dll   [+] ole32.dll     [+] oleacc.dll
28790: [+] oleaut32.dll   [+] oledlg.dll   [+] opengl32.dll
28791: [+] shell32.dll   [+] shlwapi.dll   [+] user32.dll
28792: [+] usp10.dll     [+] version.dll   [+] winhttp.dll
28793: [+] wininet.dll   [+] winmm.dll     [+] winspool.drv   [+] KERNEL32.DLL
28794: [+] ws2_32.dll    [+] wsck32.dll    [+] msimg32.dll   [+] netapi32.dll
28795:
28796: Ref:
28797: https://www.virustotal.
      com/en/file/1bc67cfe09ec7ba86ef4e8015dae1380c2ed243bf8876738e3ab809d65093bb2/analysis/1454694234/
28798:
28799: Old Ref:
28800: https://www.virustotal.
      com/en/file/c51abbc4533c2a13430ecec4efc37857d173476ccb0115b4d91079227239e59a/analysis/1454793082/
28801:
28802: MD5 8d27e8b32577839e9965903a5f4f4a8c
28803: SHA1 cdc0d39fe16ce883ea98ff65c7e31c874fe1520b
28804: SHA256 c51abbc4533c2a13430ecec4efc37857d173476ccb0115b4d91079227239e59a
28805: ssdeep 393216:59jaUQounaHiLdx3J12gpZZA0K4fvu7JwqA1KWR4LZTo:5r/IvLf2GnFK4+9d8mL
28806:
28807: authentihash 3eb14eb738e02795fb6047f2bff438535166298deff9171d2f9a214c9623c89
28808: imphash 180563e72c1153f8386aa88e7d0e59ed
28809: File size 25.3 MB ( 26506752 bytes )
28810: File type Win32 EXE
28811: Magic literal PE32 executable for MS Windows (GUI) Intel 80386 32-bit
28812: TrID Windows ActiveX control (69.3%)
28813: InstallShield setup (25.6%)
28814: Win32 Executable (generic) (2.6%)
28815: Generic Win/DOS Executable (1.1%)
28816: DOS Executable Generic (1.1%)
28817:
28818: Target machine Intel 386 or later processors and compatible processors
28819: Compilation timestamp 2016-02-06 19:03:09
28820: Link date 8:03 PM 2/6/2016
28821: Entry Point 0x013D29C8
28822: Number of sections 10
28823:
28824:
28825: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3124263) .
28826: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3124262) .
28827: Update for Windows 10 Version 1511 for x64-based Systems (KB3140741)
28828:
28829: Latest Ref:
28830: https://www.virustotal.
      com/en/file/584ca53d6dd8f7de17d0a1959bf78aad04697cf0ad7b3f23aee90b5ff720ede1/analysis/1460194451/
28831: https://www.virustotal.
      com/en/file/a17988574c08c464a59b88732ce93e58b96af65fc15d9010a35c19d454932c3/analysis/1461849317/
28832: https://www.virustotal.
      com/en/file/29e13fce486682a53750098add05ed42bc47974115f2df237ba3a6f5db374d5c/analysis/1463932733/
28833:
28834: https://www.virustotal.
      com/en/file/29a550d2ccb52ab745e66fd0ba3d7851e734956a4c522db36230d6ca528ba085/analysis/1474627883/
28835:
28836: SHA256: 29a550d2ccb52ab745e66fd0ba3d7851e734956a4c522db36230d6ca528ba085
28837: File name: maXbox4.exe
28838: Detection ratio: 0 / 57
28839: Analysis date: 2016-09-23 10:51:23 UTC ( 2 minutes ago )
28840:
28841: MD5 d958f67530da12990f09cfad7bleea16
28842: SHA1 17c3a5f20f7474acecc5565832773624eeccabc1
28843: SHA256 29a550d2ccb52ab745e66fd0ba3d7851e734956a4c522db36230d6ca528ba085
28844: ssdeep 393216:1301SCVu5Lx61ELvI9UaeUKMAtG/8+ph719oe++f5S9/XA:14SaULx6K8Ajgh0+hS1
28845:
28846: authentihash e9a70a88f590c63bdffdeal58e403de2f2121b8df3eadcc2fc57bca9b831e5df
28847: imphash 1dbc0ab427742791c686fea096e65d00
28848: File size 26.3 MB ( 27554304 bytes )
28849: File type Win32 EXE
28850: Magic literal PE32 executable for MS Windows (GUI) Intel 80386 32-bit
28851: ****
28852: Microsoft Windows SDK for Windows 7 and .NET Framework 4 Interface (UI Automation)
28853: Overall test results Number of tests 68 Verify
28854: Passed 67
28855: Failed 1
28856: Unexpected Error 0
28857:
28858: UI Tests caused Unexpected Error: 0
28859:
28860: Tests Failed: 1
28861: Test Name : ControlType.Window:SetFocus.1
28862: Result Status: Failed

```

```
28863: Test Priority: Pr1
28864: Description: Precondition: Some controls are not persisted after they loose focus (ie, Floating edit in Excel), don't tests these
28865:
28866: !! Exception !!
28867: Message: Step 12 : Focus event was not fired and was expected to be fired
28868: Unexpected: false Known Bug: false Incorrect Configuration: false
28869:
28870: Stack Trace: at InternalHelper.Tests.TestObject.ThrowMe(CheckType checkType, Exception exceptionThrown, String format, Object[] args) at InternalHelper.Tests.TestObject.TestIfEventShouldFire(EventFired shouldFire, EventFired actualFired, Object eventId, CheckType checkType) at InternalHelper.Tests.TestObject.TSC_VerifyFocusedChangeEvent(AutomationElement element, EventFired shouldFire, String eventHandlerVar, CheckType checkType) at Microsoft.Test.UIAutomation.Tests.Patterns.AutomationElementTests.TS_VerifyFocusChangeEventHelper(String EventHandlerVar) at Microsoft.Test.UIAutomation.Tests.Patterns.AutomationElementTests.TestSetFocus1a(TestCaseAttribute testCaseAttribute)
28871:
28872: Tests Passed:67/68
28873: Test Name : ControlType.Window:FocusedElement.1
28874: Test Name : ControlType.Window:GetRuntimeIdType.1
28875: Test Name : ControlType.Window:GetSupportedPatterns.1
28876: Test Name : ControlType.Window:GetSupportedProperties.1
28877: Test Name : ControlType.Window:GetType.1
28878: Test Name : ControlType.Window:HwndWindowRect.1.MSAA
28879: Test Name : ControlType.Window:HwndWindowRect.2.UIA
28880: Test Name : ControlType.Window:LocalizedControlType.1.8.1
28881: Test Name : ControlType.Window:ToString.1
28882: Test Name : ControlType.Window:AutomationIdProperty.1.7.1
28883: Test Name : ControlType.Window:AutomationIdProperty.1.7.8
28884: Test Name : ControlType.Window:BoundingRect.1
28885: Test Name : ControlType.Window:ClickablePointProperty.1
28886: Test Name : ControlType.Window:ControlTypeProperty.1
28887: Test Name : ControlType.Window:GetClickablePoint.2
28888: Test Name : ControlType.Window:GetHashCode.1
28889: Test Name : ControlType.Window:IsKeyboardFocusable.1.1.1
28890: Test Name : ControlType.Window:IsPasswordProperty.1.3.2
28891: Test Name : ControlType.Window:KeyboardFocusable.1.1.2
28892: Test Name : ControlType.Window:Navigation.1
28893: Test Name : ControlType.Window:Navigation.10
28894: Test Name : ControlType.Window:Navigation.11
28895: Test Name : ControlType.Window:Navigation.2
28896: Test Name : ControlType.Window:Navigation.3
28897: Test Name : ControlType.Window:Navigation.4
28898: Test Name : ControlType.Window:Navigation.5
28899: Test Name : ControlType.Window:Navigation.6
28900: Test Name : ControlType.Window:Navigation.7
28901: Test Name : ControlType.Window:Navigation.8
28902: Test Name : ControlType.Window:Navigation.9
28903: Test Name : ControlType.Window:Navigation.ControlFirstChild.1
28904: Test Name : ControlType.Window:Navigation.ControlLastChild.1
28905: Test Name : ControlType.Window:Navigation.ControlNextSibling.1
28906: Test Name : ControlType.Window:Navigation.ControlParent.1
28907: Test Name : ControlType.Window:Navigation.ControlPreviousSibling.1
28908: Test Name : ControlType.Window:SetFocus.10
28909: Test Name : ControlType.Window:SetFocus.3
28910: Test Name : ControlType.Window:AutomationElement.PropertyChange.Enabled.1
28911: Test Name : ControlType.Window:AutomationElement.PropertyChange.Name.1
28912: Test Name : ControlType.Window:GetCurrentPattern.1
28913: Test Name : ControlType.Window:GetCurrentPropertyValue.1
28914: Test Name : ControlType.Window:SetFocus.4
28915: Test Name : ControlType.Window:BulkAdd.1
28916: Test Name : ControlType.Window:BulkRemove.1
28917: Test Name : ControlType.Window:ChildAdd.1
28918: Test Name : ControlType.Window:ChildRemove.1
28919: Test Name : ControlType.Window:Invalidate.1
28920: Test Name : ControlType.Window:Reorder.1
28921: Test Name : ControlType.Window:TestContentView
28922: Test Name : ControlType.Window:TestControlPatterns
28923: Test Name : ControlType.Window:TestControlProperties
28924: Test Name : ControlType.Window:TestControlView
28925: Test Name : ControlType.Window:CanRotate.1.6
28926: Test Name : ControlType.Window:Move.2.1
28927: Test Name : ControlType.Window:Move.2.2
28928: Test Name : ControlType.Window:Move.2.8
28929: Test Name : ControlType.Window:Move.2.13
28930: Test Name : ControlType.Window:Move.2.17
28931: Test Name : ControlType.Window:Move.2.18
28932: Test Name : ControlType.Window:Move.2.19
28933: Test Name : ControlType.Window:Move.2.3
28934: Test Name : ControlType.Window:Move.2.4
28935: Test Name : ControlType.Window:Move.2.5
28936: Test Name : ControlType.Window:Window.MaximizableProperty.S.6.1
28937: Test Name : ControlType.Window:Window.MaximizableProperty.S.6.2
28938: Test Name : ControlType.Window:Window.MinimizableProperty.S.7.2
28939: Test Name : ControlType.Window:Window.ModalProperty.S.10.1
28940:
28941:
28942: *****
```

```

28943: Release Notes maXbox 4.2.4.80 October 2016 Ocean7 mX4
28944: ****
28945: add 20 units + 442 functions - WMI Script Type Library - webbox
28946:
28947: 1241 uPSI_wmiserv.pas {uPSI_SimpleSFTP.pas}
28948: 1242 uPSI_WbemScripting_TLB.pas
28949: 1243 unit uPSI_uJSON2;
28950: 1244 uPSI_RegSvrUtils.pas
28951: 1245 unit uPSI_osFileUtil;
28952: 1246 unit uPSI_SHDocVw; //TWebbrowser
28953: 1247 unit uPSI_SHDocVw_TLB;
28954: 1248 uPSC_classes.pas V2
28955: 1249 uPSR_classes.pas V2
28956: 1250 uPSI_U_Oscilloscope4_2
28957: 1251 unit uPSI_xutils.pas
28958: 1252 uPSI_ietf.pas
28959: 1253 uPSI_iso3166.pas
28960: 1254 uPSI_dateutil_real.pas //Optima ISO 8601
28961: 1255 unit uPSI_dateext4.pas
28962: 1256 uPSI_locale.pas
28963: 1257 file charset.inc //IANA Registered character sets
28964: 1258 unit uPSI.Strings;
28965: 1259 unit uPSI_crc_checks; //ISO 3309 and ITU-T-V42
28966: 1260 unit uPSI_extDOS;
28967:
28968: SHA1: of 4.2.4.80 15565A557B0F9576AA5F23F2A1D06BE9699A757B
28969: CRC32: FA1F1F25 26.4 MB (27,720,144 bytes)
28970:
28971: function GetCachedFileFromURL(strUL: string; var strLocalFile: string): boolean;
28972: function IAddrToHostName(const IP: string): string;
28973: function GetIEHandle(WebBrowser: TWebbrowser; ClassName: string): HWND;
28974: function GetTextFromHandle(WinHandle: THandle): string;
28975: procedure Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
28976: function FillWebForm(WebBrowser:TWebBrowser;FieldName:string;Value:string):Bool;
28977: procedure WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode: string);
28978: function NetSend(dest, Source, Msg: string): Longint; overload;
28979: function RecordsetFromXML2(const XML: string): variant;';
28980: function RecordsetToXML2(const Recordset: variant): string';
28981: Function GetCharEncoding( alias : string; var _name : string ) : integer';
28982: Function MicrosoftCodePageToMIMECharset( cp : Word ) : string';
28983: Function MicrosoftLanguageCodeToISOCODE( langcode : integer ) : string';
28984: procedure CopyHTMLToClipboard(const str: string; const htmlStr: string = '');
28985: function RFC1123ToDateTime(Date: string): TDateTime;
28986: function DateTimeToRFC1123(aDate: TDateTime): string;
28987: procedure CopyHTMLToClipboard(const str: string; const htmlStr: string);
28988: procedure DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);
28989: procedure DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);
28990: procedure DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);
28991: Function checkSystem: string;
28992: Function getSystemReport: string;
28993:
28994: ****
28995: Release Notes maXbox 4.2.4.25 June 2016 Ocean5 mX4
28996: ****
28997: add 16 units and 225 functions - Class Helper - KMemo RTF -DOSOutput
28998:
28999: 1224 uPSI_IdAntiFreeze.pas
29000: 1225 uPSI_IdLogStream.pas
29001: 1226 unit uPSI_IdThreadSafe;
29002: 1227 unit uPSI_IdThreadMgr;
29003: 1228 unit uPSI_IdAuthentication;
29004: 1229 unit uPSI_IdAuthenticationManager;
29005: 1230 uPSI_OverbyteIcsConApp
29006: 1231 unit uPSI_KMemo;
29007: 1232 unit uPSI_OverbyteIcsTicks64;
29008: 1233 unit uPSI_OverbyteIcsShal.pas
29009: 1234 unit uPSI_KEditCommon.pas
29010: 1235 unit uPSI_UtilsMax4.pas
29011: 1236 unit uPSI_IdNNTPServer;
29012: 1237 unit uPSI_UWANTUtils;
29013: 1238 unit uPSI_UtilsMax5.pas;
29014: 1239 unit uPSI_OverbyteIcsAsnlUtils;
29015: 1240 unit uPSI_IdHTTPHeaderInfo; //mX response headers
29016:
29017: SHA1: of 4.2.4.25 A52ACF844808285D8EE978637365B74B3C7C342F
29018: CRC32: CB882FFC 26.0 MB (27,276,288 bytes)
29019:
29020: ****
29021: Release Notes maXbox 4.2.2.95 Mai 2016 Ocean3 mX4
29022: ****
29023: add 10 units and 45 functions - Class Helper - KMemo RTF
29024:
29025: 1224 uPSI_IdAntiFreeze.pas
29026: 1225 uPSI_IdLogStream.pas
29027: 1226 unit uPSI_IdThreadSafe;
29028: 1227 unit uPSI_IdThreadMgr;
29029: 1228 unit uPSI_IdAuthentication;
29030: 1229 unit uPSI_IdAuthenticationManager;
29031: 1230 uPSI_OverbyteIcsConApp

```

```

29032: 1231 unit uPSI_KMemo;
29033: 1232 unit uPSI_OverbyteIcsTicks64;
29034: 1233 unit uPSI_OverbyteIcssShal.pas
29035: 1234 unit uPSI_KEditCommon.pas
29036:
29037: SHA1: of maXbox4.exe (4.2.2.95) 3EBECE1B08602BBC2785B1C9B8827EECF092330
29038: CRC32: 21D1039C, Size of EXE: 27,179,008
29039:
29040: function ContentTypeGetExtn(Const Content: String; var CLSID: String): string;
29041: function ContentTypeFromExtn(Const Extension: String): string;
29042: function DateTimeDiff2(Start, Stop : TDateTime) : int64;
29043: Function LoadStringOfFile( const AFile : string ) : string';
29044: Function LSOF( const AFile : string ) : string';
29045:
29046: ****
29047: Release Notes maXbox 4.2.2.90 April 2016 Ocean2 mX4
29048: ****
29049: add 10 units and 15 functions - minor bugfixes
29050: http://max.kleiner.com/boxart.htm
29051:
29052: 1212 unit uPSI_MapFiles.pas //map stream of memory-mapped files
29053: 1213 unit uPSI_BKPwdGen, //Password Generator
29054: 1214 unit uPSI_Kronos, // big chrono date time library
29055: 1215 unit uPSI_TokenLibrary2;
29056: 1216 uPSI_KDialogs,
29057: 1217 uPSI_Numedit,
29058: 1218 unit uPSI_StSystem2;
29059: 1219 unit uPSI_KGraphics;
29060: 1220 uPSI_KGraphics_functions;
29061: 1221 uPSI_umaxPipes.pas
29062:
29063:
29064: ****
29065: Release Notes maXbox 4.2.0.80 April 2016 Ocean mX4
29066: ****
29067:
29068: This is an upgrade to mX3 app files dir, if you want all examples/docs for this mX4
29069: you have to download mX3 first and then copy mX4 files in it too.
29070: Otherwise you already own mX3 on disk just copy new files (save maxboxdef.ini first).
29071: All functions & object: maxbox_functions_all.pdf
29072: News: Add 5 Units, 1 Tutor, Pipe Library2, KLog, FPlot42
29073:
29074: 1207 unit uPSI_CPUSpeed.pas
29075: 1208 uPSI_RoboTracker.pas
29076: 1209 unit uPSI_NamedPipesImpl.pas
29077: 1210 unit uPSI_KLog.pas
29078: 1211 unit uPSI_NamedPipeThreads.pas
29079:
29080:
29081: //Commonly used Delphi WinAPI routines
29082: http://www.rosseeld.be/DRO/Delphi/Delphi%20WinAPI.htm
29083:
29084: ****
29085: Release Notes maXbox 4.2.0.10 March 2016 Ocean mX4
29086: ****
29087:
29088: News:
29089: Add 9 Units, 1 Tutor, InetUtils, Chron, REXX funcs
29090:
29091: 1198 unit uPSI_Simpat;
29092: 1199 unit uPSI_Tooltips.pas
29093: 1200 unit uPSI_StringGridLibrary.pas
29094: 1201 unit uPSI_ChronCheck.pas
29095: 1202 unit uPSI_REXX.pas
29096: 1203 uPSI_SysImg.pas
29097: 1204 unit uPSI_Tokens;
29098: 1205 unit uPSI_KFunctions;
29099: 1206 unit uPSI_KMessageBox;
29100:
29101: ****
29102: Release Notes maXbox 4.0.2.80 Feb 2016 Jupiter mX4
29103: ****
29104:
29105: This is an upgrade to mX3 app files dir, if you want all examples/docs for this mX4
29106: you have to download mX3 first and then copy mX4 files in it too.
29107: Otherwise you already have mX3 on disk just copy the new files (save maxboxdef.ini first).
29108:
29109: News:
29110: Add 75 Units, 12 Tutors, ChangeTracker, CPP+, OLEUtils2, xmldom, Chess4, 3DFrame, XMLRPC, X509
29111:
29112: 1121 unit uPSI_IndySockTransport.pas (+IdHTTPHeaderInfo)
29113: 1122 unit uPSI_HTTPProd.pas
29114: 1123 unit uPSI_CppParser.pas
29115: 1124 unit uPSI_SynHighlighterCpp.pas
29116: 1125 unit uPSI_CodeCompletion.pas
29117: 1126 unit uPSI_U_IntList2.pas
29118: 1127 unit uPSI_SockHTTP.pas
29119: 1128 uPSI_SockAppNotify.pas
29120: 1129 uPSI_NSToIS.pas

```

```

29121: 1130 unit uPSI_DBOleCtl.pas
29122: 1131 unit uPSI_xercesxmldom;
29123: 1132 unit uPSI_xmldom;
29124: 1133 unit uPSI_Midas;
29125: 1134 unit uPSI_JclExprEval;
29126: 1135 uPSI_Gameboard;
29127: 1136 unit uPSI_ExtUtil;
29128: 1137 unit uPSI_FCGIApp;
29129: 1138 unit uPSI_ExtPascal;
29130: 1139 unit uPSI_PersistSettings;
29131: 1140 IdHTTPHeaderInfo.pas
29132: 1141 uPSI_SynEditAutoComplete;
29133: 1142 uPSI_SynEditTextBuffer.pas
29134: 1143 unit uPSI_JclPCRE;
29135: 1144 unit uPSI_ZConnection;
29136: 1145 unit uPSI_ZSequence;
29137: 1146 unit uPSI_ChessPrg;
29138: 1147 unit uPSI_ChessBrd;
29139: 1148 unit uPSI_Graph3D;
29140: 1149 uPSI_SysInfoCtrls2.pas
29141: 1150 unit uPSI_RegUtils;
29142: 1151 unit uPSI_VariantRtn;
29143: 1152 uPSI_StdFuncs,
29144: 1153 unit uPSI_SQLTxtRtns;
29145: 1154 unit uPSI_BSpectrum;
29146: 1155 unit IPAddressControl;
29147: 1156 unit uPSI_Paradox;
29148: 1157 unit uPSI_Environ;
29149: 1158 uPSI_GraphicsPrimitivesLibrary;
29150: 1159 uPSI_DrawFigures,
29151: 1160 unit uPSI_synadbg;
29152: 1161 unit uPSI_BitStream;
29153: 1162 unit uPSI_xrtl_util_FileVersion;
29154: 1163 uPSI_XmlRpcCommon,
29155: 1164 unit uPSI_XmlRpcClient;
29156: 1165 unit uPSI_XmlRpcTypes;
29157: 1166 unit uPSI_XmlRpcServer;
29158: 1167 unit uPSI_SynAutoIndent;
29159: 1168 unit uPSI_synafpc;
29160: 1169 unit uPSI_RxNotify;
29161: 1170 unit uPSI_SynAutoCorrect;
29162: 1171 unit uPSI_rxOle2Auto;
29163: 1172 unit uPSI_Spring_Utilsmx;
29164: 1173 unit uPSI_ulogifit;
29165: 1174 unit uPSI_HarmFade;
29166: 1175 unit uPSI_SynCompletionProposal;
29167: 1176 unit uPSI_rxAniFile;
29168: 1177 uPSI_ulinfif,
29169: 1178 uPSI_usvdif;
29170: 1179 unit uPSI_JclStringLists;
29171: 1180 unit uPSI_ZLib;
29172: 1181 unit uPSI_MaxTokenizers; //WANT
29173: 1182 unit uPSI_MaxStrUtils;
29174: 1183 unit uPSI_MaxXMLUtils;
29175: 1184 unit uPSI_MaxUtils;
29176: 1185 unit uPSI_VListBox;
29177: 1186 unit uPSI_MaxDOM;
29178: 1187 unit uPSI_MaxDOMDictionary;
29179: 1188 unit uPSI_MaxDOMDictionary_Routines;
29180: 1189 unit uPSI_cASN1;
29181: 1190 unit uPSI_cX509Certificate;
29182: 1191 unit uPSI_uCiaXml;
29183: 1192 unit uPSI.StringsW;
29184: 1193 unit uPSI_FileStreamW; //WideString
29185: 1194 unit Drawingutils;
29186: 1195 unit uPSI_InetUtilsUnified;
29187: 1196 unit uPSI_FileMask;
29188: 1197 unit uPSI_StrConv;
29189:
29190: SHA1: of maXbox4.exe (4.0.2.80) CDC0D39FE16CE883EA98FF65C7E31C874FE1520B
29191: CRC32: 64E170B0
29192:
29193: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
29194:
29195: ****
29196: Release Notes maXbox 3.9.9.195 Mai 2015 CODEsign
29197: ****
29198: Add 36 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, Wave ACM
29199: Refactor DB Constructor - Virtual Constructor - Override
29200:
29201: 1085 unit uPSI_JvAnimate          //JCL
29202: 1086 unit uPSI_DBXCharDecoder;   //DBX
29203: 1087 unit uPSI_JvDBLists;       //JCL
29204: 1088 unit uPSI_JvFileInfo;      //JCL
29205: 1089 unit uPSI_SOAPConn;        //VCL
29206: 1090 unit uPSI_SOAPLinked;     //VCL
29207: 1091 unit uPSI_XSBuiltIns;     //VCL
29208: 1092 unit uPSI_JvgDigits;      //JCL
29209: 1093 unit uPSI_JvDesignUtils;
```

```

29210: 1094 unit uPSI_JvgCrossTable;
29211: 1095 unit uPSI_JvgReport;      1096 unit uPSI_JvDBRichEdit;
29212: 1097 unit uPSI_JvWinHelp;
29213: 1098 unit uPSI_WaveConverter;  1099 unit uPSI_ACMConvertor;
29214: 1100 unit XSBuiltIns_Routines
29215: 1101 unit uPSI_ComObjOleDB_utils.pas
29216: 1102 unit uPSI_SMScript;
29217: 1103 unit uPSI_CompFileIo;
29218: 1104 unit uPSI_SynHighlighterGeneral;
29219: 1105 unit uPSI_geometry2;
29220: 1106 unit uPSI_MConnect
29221: 1107 unit uPSI_ObjBrkr;
29222: 1108 unit uPSI_uMultiStr;
29223: 1109 unit uPSI_WinAPI.pas;
29224: 1110 unit uPSI_JvAVICapture;
29225: 1111 unit uPSI_JvExceptionForm;
29226: 1112 unit uPSI_JvConnectNetwork;
29227: 1113 unit uPSI_MainForm;
29228: 1114 unit uPSI_DdeMan;
29229: 1115 unit uPSI_DIUtils;
29230: 1116 unit uPSI_gnugettext;
29231: 1117 unit uPSI_XmlXform;
29232: 1118 unit uPSI_SvrHTTPIndy;
29233: 1119 unit uPSI_CPortTrmSet;
29234: 1120 unit SvrLog;
29235:
29236: SHA1: maXbox3.exe F0AB7D054111F5CE46BA122D6280397A841C6FAB
29237: CRC32: maXbox3.exe 602A885C
29238:
29239: ****
29240: Release Notes maXbox 3.9.9.180 Feb 2015 CODEsign
29241: ****
29242: Add 20 Units, 1 Slide, Tutor, Big Numbers, TestFramework, GEOInfo
29243:
29244: 1065 unit uPSI_UDict;           //DDF
29245: 1066 unit uPSI_ubigFloatV3;    //DDF
29246: 1067 unit uPSI_UBigIntsV4;     //DDF
29247: 1068 unit uPSI_ServiceMgr2;    //mX
29248: 1069 unit uPSI_UP10Build;      //PS
29249: 1070 unit uPSI_UParser10;      //PS
29250: 1071 unit uPSI_IdModBusServer; //MB
29251: 1072 unit uPSI_IdModBusClient; +MBUtils //MB
29252: 1073 unit uPSI_ColorGrd;       //VCL
29253: 1074 unit uPSI_DirOutln;       //VCL
29254: 1075 unit uPSI_Gauges;        //VCL
29255: 1076 unit uPSI_CustomizeDlg;   //VCL
29256: 1077 unit uPSI_ActnMan;       //VCL
29257: 1078 unit uPSI_CollPanl;       //VCL
29258: 1079 unit uPSI_Calendar2;     //VCL
29259: 1080 unit uPSI_IBCtrls;       //VCL
29260: 1081 unit uPSI_IdStackWindows; //Indy
29261: 1082 unit uPSI_CTSVendorUtils;
29262: 1083 unit uPSI_VendorTestFramework;
29263: 1084 unit uPSI_TInterval;
29264:
29265: SHA1: maXbox3.exe 3D7F88BE9687CB834A5E2DAED08B23358484FEC0
29266: CRC32: maXbox3.exe E2ADE828
29267:
29268: ****
29269: Release Notes maXbox 3.9.9.160 January 2015 CODEsign
29270: ****
29271: Add 9 Units, 2 Slides 1 Tutor, CLXUp, ExampleEdition, UnitConverter
29272: ExecuteProcess (MultiProcessor), ConsoleCapture (DOS)
29273:
29274: 1053 unit uPSI_BigIni;          //Hinzen
29275: 1054 unit uPSI_ShellCtrls;     //VCL
29276: 1055 unit uPSI_fMath;         //FMath
29277: 1056 unit uPSI_fComp;         //FMath
29278: 1057 unit uPSI_HighResTimer;  //Lauer
29279: 1058 unit uconvMain; (Unit Converter) //PS
29280: 1059 unit uPSI_uconvMain;     //PS
29281: 1060 unit uPSI_ParserUtils;  //PS
29282: 1061 unit uPSI_upSUtils;      //PS
29283: 1062 unit uPSI_ParserU;       //PS
29284: 1063 unit uPSI_TypeInfo;      //VCL
29285: 1064 unit uPSI_ServiceMgr;   //mX
29286:
29287: ****
29288: Release Notes maXbox 3.9.9.120 December 2014 CODEsign
29289: ****
29290: Add 10 Units, 1Slides, NeuralNetwork, Pan3D View, GDIBackend
29291:
29292: 1043 unit uPSI_NeuralNetwork;
29293: 1044 unit uPSI_StExpr;
29294: 1045 unit uPSI_GR32_Geometry;
29295: 1046 unit uPSI_GR32_Containers;
29296: 1047 unit uPSI_GR32_Backends_VCL,
29297: 1048 unit uPSI_StSaturn; //Venus+Pluto+Mars+Merc+JupSat+ ++
29298: 1049 unit uPSI_JclParseUses;

```

```
29299: 1050 unit uPSI_JvFinalize;
29300: 1051 unit uPSI_panUnit1;
29301: 1052 unit uPSI_DD83u1; //Arduino Tester
29302:
29303:
29304:
29305: Published Doc maxbox Tutors Starter Introduction 2014
29306: Tutorial 00 Blix the Programmer
29307: Tutorial 01 Procedural-Coding
29308: Tutorial 02 OO-Coding
29309: Tutorial 03 Modular Coding
29310: Tutorial 04 UML Coding
29311: Tutorial 05 Internet Coding
29312: Tutorial 06 Network Coding
29313: Tutorial 07 Game Coding
29314: Tutorial 08 Operating System Coding
29315: Tutorial 09 Database Coding
29316: Tutorial 10 Statistic Coding
29317: Tutorial 10 Probability Coding
29318: Tutorial 11 Forms Coding
29319: Tutorial 12 SQL Coding
29320: Tutorial 13 Crypto Coding
29321: Tutorial 14 Parallel Coding
29322: Tutorial 15 Serial Coding
29323: Tutorial 16 Event Driven Coding
29324: Tutorial 17 Web Server Coding
29325: Tutorial 18 Arduino Coding and Web of Things
29326: Tutorial 18_3 Arduino RGB LED Breadboard and Source LED zip
29327: Tutorial 19 WinCOM /Arduino Coding and Source LED COM
29328: Tutorial 20_1 Regular Expressions V2
29329: Tutorial 21 Android av. End of 2014 and Basta LED Things and Code ADK SeekBar
29330: Tutorial 22 Services Coding
29331: Tutorial 23 Real Time Code
29332: Tutorial 24 Clean Code
29333: Tutorial 25 Configuration
29334: Tutorial 26 TCP Sockets
29335: Tutorial 27 XML Coding
29336: Tutorial 28 DLL Coding
29337: Tutorial 29 UML Modeling
29338: Tutorial 30 WOT Web of Things and Basta 2014 Arduino and maxbox
29339: Tutorial 31 Closures
29340: Tutorial 32 SQL Server Firebird
29341: Tutorial 33 Oscilloscope
29342: Tutorial 34 GPS Codes
29343: Tutorial 35 Web Box
29344: Tutorial 36 Function Testing
29345: Tutorial 37 API Coding
29346: Tutorial 38 3D Coding (2015)
29347: Tutorial 39 Maps Coding
29348: Tutorial 39 Maps2 Coding
29349: Tutorial 39 GEO Map Coding (2015)
29350: Tutorial 39_1 GEO Map OpenLayers (2015)
29351: Tutorial 40 REST Coding (coming 2015)
29352: Tutorial 41 Big Numbers Coding (coming 2015)
29353: Tutorial 42 Parallel Processing (coming 2015)
29354: Tutorial 43 Code Metrics: June2016
29355: Tutorial 44 IDE Code Extensions
29356: http://www.softwareschule.ch/download/maxbox_starter44.pdf
29357: Tutorial 45 Robotics: July2016 available
29358:
29359: http://www.slideshare.net/maxkleiner1/codesign-2015
29360: http://basta_2015_speaker_max_kleiner
29361: http://www.arduino.cc/en/Tutorial/ASCIITable
29362: http://www.vwlowen.co.uk/arduino/usb-digital/pc-control.htm
29363: http://www.yunqa.de/delphi/doku.php/products/regex/syntax#quantifiers
29364: http://elib.uni-stuttgart.de/opus/volltexte/2008/3440/pdf/diss_kroell_hs.pdf
29365: http://www.softwareschule.ch/download/exampleedition2016.zip
29366:
29367:
29368: Published Doc maxbox Example Edition 2015
29369:
29370: \example_edition\01_Algorithm';
29371: \example_edition\02_Graphics';
29372: \example_edition\03_Games';
29373: \example_edition\04_Multimedia';
29374: \example_edition\05_Internet';
29375: \example_edition\06_Communication';
29376: \example_edition\07_Geographical';
29377: \example_edition\08_Operating';
29378: \example_edition\09_Database';
29379: \example_edition\10_Science';
29380: \example_edition\11_EMBEDDED';
29381: \example_edition\12_Security';
29382: \example_edition\13_General';
29383: \example_edition\14_Energy';
29384: \example_edition\15_Transport';
29385: \example_edition\16_Robotics';
```

```

29388: Ref:
29389:   if IsValidPeFile(exepath+'maxbox3.exe') then begin
29390:     x1:= ComputePEChecksum(exepath+'maxbox3.exe'); // original filename
29391:     x2:= ComputePEChecksum(exepath+'maxbox3.exe');
29392:   end;
29393:   WriteLn('Checksum 1: '+ itoa(x1)+ #13#10+'Checksum 2: '+ itoa(x2));
29394:
29395:   if ConnectDrive('Z:','\MAXBOX8\Users\Public', True,True) = NO_ERROR then
29396:     writeln('Net Share Z:\ Connected');
29397:     DisconnectNetDrive('Z:', True, True, True);
29398:
29399:   ComTerminal:= TCustomComTerminal.create(self);
29400:   EditComTerminal(comterminal); //TComTrmSetForm comterminal.Free;
29401:   LastSysErrorMessage : string; LastSysErrorMessageA : AnsiString';
29402:
29403:   writeln(shaltohex(SHA1ofStr(loadstringj(exepath+'maxbox4.exe'))))
29404:   writeln(shaltohex(SHA1ofStr(loadstringoffile(exepath+'maxbox4.exe'))))
29405:   writeln(strtoHexl(SHA1ofStr(lsof(exepath+'maxbox4.exe'))))
29406:   writeln(shaltohex(SHA1ofStr(fileToString(exepath+'maxbox4.exe'))))
29407:
29408: PE Checksum 1: 27724732 PE Checksum 2: 27724732
29409: PE Checksum 1: 27767976 PE Checksum 2: 27767976
29410: SHA1: 15565A557B0F9576AA5F23F2A1D06BE9699A757B
29411:
29412: Verifying: C:\maxbox\maxbox3\work2015\maXbox3digisign_certificates\maXbox4sign.exe
29413:
29414: C:\maxbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maxbox4.exe
29415:
29416: Verifying: maXbox4.exe
29417: Signature Index: 0 (Primary Signature)
29418: Hash of file (sha1): 02BF5152206C9AC252B47B33BC4D63BC9354227A
29419:
29420: Signing Certificate Chain:
29421:   Issued to: maXboxCertAuth
29422:   Issued by: maXboxCertAuth
29423:   Expires: Sun Jan 01 01:59:59 2040
29424:   SHA1 hash: 6F83207B500DCC0E32A719599C9C6BD7E6B2A04D
29425:
29426:   Issued to: maXbox4signer
29427:   Issued by: maXboxCertAuth
29428:   Expires: Sun Jan 01 01:59:59 2040
29429:   SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
29430:
29431:   Issued to: maXbox4exe
29432:   Issued by: maXbox4signer
29433:   Expires: Sun Jan 01 01:59:59 2040
29434:   SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
29435:
29436: The signature is timestamped: Fri Oct 14 21:52:01 2016
29437: Timestamp Verified by:
29438:   Issued to: Starfield Root Certificate Authority - G2
29439:   Issued by: Starfield Root Certificate Authority - G2
29440:   Expires: Fri Jan 01 01:59:59 2038
29441:   SHA1 hash: B51C067CEE2B0C3DF855AB2D92F4FE39D4E70F0E
29442:
29443:   Issued to: Starfield Timestamp Authority - G2
29444:   Issued by: Starfield Root Certificate Authority - G2
29445:   Expires: Tue Feb 16 09:00:00 2021
29446:   SHA1 hash: 8D517FC99F3B23CEC36B3B0CB3B7E51ACD344BCE
29447:
29448: Successfully verified: maXbox4.exe
29449:
29450: Number of files successfully Verified: 1
29451: Number of warnings: 0
29452: Number of errors: 0
29453:
29454: C:\maxbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
29455: C:\maxbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
29456:
29457: Amount of Functions: 17622
29458: Amount of Procedures: 10532
29459: Amount of Constructors: 1684
29460: Totals of Calls: 29838
29461: SHA1: of 4.2.4.80 E101F7CA8FEBECCED8DA84B187B9CFB09A4D6BD5
29462: CRC32: DEFF39B1 26.4 MB (27,689,936 bytes)
29463: 000 mX4 executed: 13/10/2016 00:10:15 Runtime: 0:7:22.271 Memload: 30% use
29464:
29465: Amount of Functions: 16507
29466: Amount of Procedures: 9872
29467: Amount of Constructors: 1627
29468: Totals of Calls: 28006
29469: SHA1: of maxbox 4.2.0.10 15671711176CCEA92AEC355B44867D48B7C54575
29470: CRC32: 1517C625
29471: maXbox4.exe 26564096 bytes
29472: https://www.virustotal.com/en/file/fdb4d155d919b19c14c6382eb2e4269072cbc9e5100cb650276c48da149a729a/analysis/1457802294/
29473:
29474: Amount of Functions: 16584
29475: Amount of Procedures: 9891

```

```
29476: Amount of Constructors: 1629
29477: Totals of Calls: 28104
29478: SHA1: of maXbox 4.2.0.10 15671711176CCEA92AEC355B44867D48B7C54575
29479: CRC32: 8C581C30
29480: https://www.virustotal.
    com/en/file/2dbbeea235d560b7bb2cefcadb05ec328e1d95f32490980894c07c1288123e13/analysis/
29481:
29482: Amount of Functions: 16627
29483: Amount of Procedures: 9922
29484: Amount of Constructors: 1633
29485: Totals of Calls: 28182
29486: SHA1: of 4.2.0.80 638E7412750AB0ECF14F2A5515BC4A2DE561EAC2
29487: CRC32: 7C91FD2A   Exe size: 26,650,112
29488:
29489: -----
29490: Amount of Functions: 16746
29491: Amount of Procedures: 10028
29492: Amount of Constructors: 1652
29493: Totals of Calls: 28426
29494: SHA1: of 4.2.2.90 1EEE461ACF78ACA461806D85488B87A4FA08F39F
29495: CRC32: BCEDACF0, Size of Exe: 26,920,960
29496: https://www.virustotal.
    com/en/file/a17988574c08c464a59b88732ce93e58b96af65fca15d9010a35c19d454932c3/analysis/1461849317/
29497:
29498: Amount of Functions: 16910
29499: Amount of Procedures: 10173
29500: Amount of Constructors: 1679
29501: Totals of Calls: 28762
29502: SHA1: of 4.2.2.95 3E8ECE1B08602BBC2785B1C9B8827EEECF092330
29503: CRC32: 21D1039C, Size of EXE: 27,179,008
29504: ||| mX4 executed: 15/05/2016 11:28:53 Runtime: 0:6:59.491 Memload: 25% use
29505:
29506: https://www.virustotal.
    com/en/file/a2496d064ff2c95e9aed1b5e637b74a1ed402f53a11df5b332235f2029306867/analysis/1463305701/
29507:
29508: Amount of Functions: 16955
29509: Amount of Procedures: 10181
29510: Amount of Constructors: 1679
29511: Totals of Calls: 28815
29512: SHA1: of 4.2.2.98 CB4EE7E0F43AF38C882492C7E111CCF45ADEDF49
29513: CRC32: 622AE6FE, 25.9 MB (27,200,512 bytes)
29514: ||| mX4 executed: 22/05/2016 15:03:56 Runtime: 0:6:59.582 Memload: 28% use
29515: ||
29516: https://www.virustotal.
    com/en/file/29e13fce486682a53750098add05ed42bc47974115f2df237ba3a6f5db374d5c/analysis/1463932733/
29517:
29518: https://www.metadefender.com/#!/results/file/cc5cb9749a42462dbaac899c1b5aa661/regular
29519:
29520: Amount of Functions: 17035
29521: Amount of Procedures: 10215
29522: Amount of Constructors: 1679
29523: Totals of Calls: 28929
29524: SHA1: of 4.2.4.25 D93C8AD41BDC7BC2F3B0920297530FEE182E4625
29525: CRC32: B7647046 26.0 MB (27,266,560 bytes) 27266560
29526: ||| mX4 executed: 04/06/2016 22:17:05 Runtime: 0:6:54.644 Memload: 24% use
29527:
29528: https://www.virustotal.
    com/en/file/b3697f1e992323b323f9894df4f2b9ce2975dae57644181b0e9d035692edc792/analysis/1465075973/
29529:
29530: Amount of Functions: 17493
29531: Amount of Procedures: 10496
29532: Amount of Constructors: 1687
29533: Totals of Calls: 29676
29534: SHA1: of 4.2.4.60 EB25559B85503AA106553B1E3167C1F5C0A2C010
29535: CRC32: D3540E5
29536: ||| mX4 executed: 21/09/2016 16:23:00 Runtime: 0:7:19.942 Memload: 32% use
29537: ||
29538:
29539: https://www.metadefender.com/#!/results/file/351c6adf098748d4b08be53c994c0693/regular/analysis
29540: https://www.virustotal.
    com/en/file/29a550d2ccbc52ab745e66fd0ba3d7851e734956a4c522db36230d6ca528ba085/analysis/1474627883/
29541:
29542: //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
29543: -----
29544: Amount of Functions: 17679
29545: Amount of Procedures: 10532
29546: Amount of Constructors: 1686
29547: Totals of Calls: 29897
29548: SHA1: of 4.2.4.80 15565A557B0F9576AA5F23F2A1D06BE9699A757B
29549: CRC32: FA1F1F25 26.4 MB (27,720,144 bytes)
29550: ||| mX4 executed: 14/10/2016 22:11:32 Runtime: 0:7:22.399 Memload: 23% use
29551: -----
29552: https://www.metadefender.com/#!/results/file/6ce34a7f1b53462daee6bdea2bad6a5/regular/analysis
29553: https://www.metadefender.com/#!/results/file/933baa9823514320806c9533482a3b89/regular/analysis
29554: https://www.virustotal.
    com/en/file/b80b0bfef22c6b4be3dbc4af984ca897144895f3c1e162f3ad7895d14fb4e667/analysis/1476479319/
29555:
29556: SHA256: b80b0bfef22c6b4be3dbc4af984ca897144895f3c1e162f3ad7895d14fb4e667
29557: File name: maXbox4.exe
```

```
29558: Detection ratio: 0 / 56
29559: Analysis date: 2016-10-14 21:08:39 UTC (3 minutes ago)
29560:
29561: MD5 97913d53d5f8b1415a13d0da74441293
29562: SHA1 15565a557b0f9576aa5f23f2a1d06be9699a757b
29563: SHA256 b80b0bfeff22c6b4be3dbc4af984ca897144895f3c1e162f3ad7895d14fb4e667
29564: ssdeep 393216:L50BtwF5+OPOQfnfzZEJWAZFkApznAtatomQ8B/Z2NXveY:LetwF5DPfdOrzdF2
29565: authentihash e35bbb36d44c5a98d878adad2e361e3a062f32a281207219869162b266939485
29566: imphash c3919d3e014cb7692d8cf1cd5e07e0e2
29567: File size 26.4 MB (27720144 bytes)
29568: File type Win32 EXE
29569: Magic literal PE32 executable for MS Windows (GUI) Intel 80386 32-bit
29570: TrID Windows ActiveX control (69.3%)
29571: InstallShield setup (25.6%)
29572: Win32 Executable (generic) (2.6%)
29573: Generic Win/DOS Executable (1.1%)
29574: DOS Executable Generic (1.1%)
29575: target machine Intel 386 or later processors and compatible processors
29576: Compilation timestamp 2016-10-14 19:37:05
29577: Entry Point 0x014C6A38
29578: Number of sections 10
29579:
29580: Overlays:
29581: MD5 b27b933d43798982316b77e0eb8d1367f
29582: File type data
29583: Offset 27716096
29584: Size 4048
29585: Entropy 7.52
29586:
29587: Name Virtual address Virtual size Raw size Entropy MD5
29588: .text 4096 21729176 21729280 6.61 356f98b528801247666ee5f434af6206
29589: .itext 21733376 52092 52224 6.60 f7dee240f8cab870b7afffc052638015d
29590: .data 21786624 406016 406016 5.26 4f6b09b46ee3b6d2e3ef32804ec08f7b
29591: .bss 22196224 386276 0 0.00 d41d8cd98f00b204e9800998ecf8427e
29592: .idata 22585344 59604 59904 5.55 92ea07271796ed1d0db4b3f780db0ffa
29593: .edata 22646784 77 512 0.89 789e31fb44f72a851873ebd95239ffe8
29594: .tls 22650880 484 0 0.00 d41d8cd98f00b204e9800998ecf8427e
29595: .rdata 22654976 24 512 0.28 ad4beafb0c82dd98ec61e878b76e7a
29596: .reloc 22659072 1368028 1368064 6.76 0d29a4d6b5701a27d53e9bfdaaaa920f
29597: .rsrc 24027136 4098560 4098560 5.39 5215e48b8da04f4c4ebcc89be7d94be2 $
29598:
29599: CodeSize 21781504
29600: InitializedDataSize 5933568
29601: TimeStamp 2016:10:14 20:37:05+01:00
29602: EntryPoint 0x14c6a38
29603: OriginalFileName maxbox4_2.exe
29604: MIMEType application/octet-stream
29605:
29606: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3140768)
29607: Update for Windows 10 Version 1511 for x64-based Systems (KB3140741)
29608: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3147458)
29609: Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421).
29610: Update for Windows 10 Version 1511 for x64-based Systems (KB3152599)
29611: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3189866).
29612: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3193494)
29613: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3194798).
29614: Windows Malicious Software Removal Tool for Windows 8, 8.1, 10 and Windows Server 2012, 2012 R2 x64 Edition - October 2016 (KB890830)
29615: ---- bigbitbox code_cleared_checked----
```